Universidad Carlos III de Madrid

Escuela Politécnica Superior

Ingeniería Técnica en Informática de Gestión

Proyecto Fin de Carrera

Diseño e Implementación de una Base de Datos para la ONG ASEM

Tutor: José María Sierra

Autor: Pablo Burgos Escribano

Abril 2009

Capítulo 1: Introducción

1.1 Introducción a las BBDD

Una base de datos es una estructura que nos permite almacenar de forma ordenada una gran cantidad de información relacionada entre sí, de manera que pueda ser procesada posteriormente con facilidad. Surge ante la necesidad de controlar y manejar grandes flujos de datos, de forma que el coste de procesarlos no supere el valor que se pueda obtener de los mismos.

En base a los documentos [1], [16] y [17], hasta que surgieron los sistemas de bases de datos actuales, la gestión de información era lenta y muy costosa, llegando a ser inviable debido a la enorme cantidad de tiempo que se requería para procesar los datos. Esa lentitud hacía que dichos datos estuvieran obsoletos antes de acabar su procesamiento. Por este motivo a finales del siglo XIX surgieron los primeros sistemas en Estados Unidos, en un primer momento su objetivo consistía en controlar el censo de la población, aunque debido a la tecnología usada (tarjetas magnéticas), los resultados tardaban años en obtenerse. Pese a ello resultaba un avance respecto a la tecnología anterior que podía tardar más del doble en procesar la misma información.

En la década de 1950 se implantaron las cintas magnéticas, que permitieron una mayor automatización y velocidad en los procesos. Aunque no duraron mucho ya que se sustituyeron por los discos magnéticos en la década de 1960. Con los discos magnéticos surgieron las que podemos considerar como las primeras bases de datos, las bases de datos en red y jerárquicas.

A partir de la década de 1970 surgió el diseño relacional, que unido al uso de discos magnéticos ofrecía una eficiencia superior a los sistemas anteriores. Gracias a eso y a su simplicidad a la hora de programar y de usar, las bases de datos relacionales se impusieron a las bases de datos en red y jerárquicas hasta finales de la década de 1980.

En la década de 1990 surge el lenguaje SQL (Structured Query Language), gracias al cual se pueden procesar enormes cantidades de datos con pocas consultas. El SQL se convierte en el estándar por excelencia en bases de datos, convirtiéndose en el más usado en SGBD comerciales en la actualidad.

En los últimos años han surgido nuevos modelos de bases de datos, como son las bases de datos deductivas, o las bases de datos orientadas a objetos. Estas últimas han surgido al calor

de los lenguajes de programación orientados a objetos y se espera que en un futuro el estándar SQL sea compatible con ellas.

Antes de centrarnos en las bases de datos relacionales, debemos plantearnos los motivos por los cuales se usan sistemas gestores de bases de datos y no sistemas de ficheros.

En un sistema de ficheros amplio, nos encontramos con varios problemas a la hora de procesar grandes cantidades de información. Por un lado las arquitecturas de 32 bits mayoritarias por el momento, no permiten direccionar más de 4GB de información, al mismo tiempo, para cada tipo de consulta es necesario elaborar una aplicación que busque en el sistema la información requerida, esas aplicaciones son complejas debido a que tienen que buscar entre un gran volumen de datos. El desarrollador debe implementar un sistema de control de concurrencia para controlar el caso de que dos usuarios modifiquen la misma información al mismo tiempo, y del mismo modo debe crear un control de transacciones, que se asegure de que las transacciones llegan a término, y en caso de error del sistema, regresa a un estado estable anterior. A todo lo anterior debemos añadir que al hacer uso de un sistema de ficheros, la seguridad depende del sistema operativo, restando versatilidad en el acceso a los datos.

Debido a todo lo anterior los sistemas de ficheros son sustituidos por bases de datos en el proceso de grandes volúmenes de información, puesto que en un SGBD actual, ya están solucionados todos estos conflictos, ofreciendo al desarrollador los recursos para acceder a los datos sin necesidad de saber de qué forma están almacenados (Independencia de datos), con una mayor eficiencia en el acceso a la información, permitiendo recuperarla y modificarla de forma segura y manteniendo la integridad y consistencia de la misma.

A todo lo anterior podemos añadir que los SGBD incluyen controles de concurrencia y de transacciones, simplificando la tarea del desarrollador, y reduciendo ampliamente el tiempo necesario para que una aplicación esté completada.

Tras todo esto, parecería que no hay motivos para no usar sistemas gestores de bases de datos en una aplicación, pero obviamente esto no es así. Debido a su complejidad, un SGBD puede resultar demasiado lento para una aplicación específica, o puede darse el caso de que dicha aplicación necesite efectuar una operación que no es soportada por el lenguaje de consultas. Por ese motivo el uso de bases de datos debe ser analizado en cada caso individualmente, siempre teniendo en cuenta las necesidades específicas de nuestra aplicación y sin descartar alternativas.

Como ya se ha indicado previamente, actualmente predominan los Sistemas Gestores de Bases de Datos Relacionales. Estos están basados en el modelo relacional, que según [6] y como su propio nombre indica, considera la base de datos como un conjunto de relaciones

(tablas), siendo cada relación un conjunto de datos (filas) donde no es relevante el orden en que se encuentren (esta es una ventaja sobre el modelo en red o jerárquico).

En una relación, cada columna representa un atributo, estos atributos poseen un dominio, es decir, solo pueden almacenar un conjunto de valores. De esta forma podemos evitar que se realicen operaciones incorrectas con los datos, si un atributo de una tabla es una cadena de caracteres de longitud 10, aunque dichos caracteres sean números, no tiene sentido efectuar operaciones matemáticas sobre ellos.

Aunque no es relevante el orden de las tuplas (filas) en una relación, si es importante evitar que estas se repitan, ya que en caso de ser así, nos encontraríamos con una colisión que provocaría inconsistencia de los datos. Por ese motivo cada tupla debe tener al menos un campo identificador o clave primaria. El valor de dicho atributo debe ser único para una tupla en la relación, de forma que permita localizar dicha tupla de forma unívoca. En caso de existir otros atributos con un valor único por cada tupla de la relación, se les tratará como claves candidatas o alternativas

Un aspecto importante del modelo relacional es el caso de las claves ajenas, se trata de atributos cuyo valor coincide con la clave primaria de otra relación (puede ser la misma relación). Las claves ajenas representan relaciones entre los datos, es decir, una tabla “Persona” puede estar relacionada con otra tabla “Coche”, si en esta última existe un atributo “propietario”, que contiene la clave primaria de la persona propietaria del coche.

Como dice en [1], toda base de datos relacional debe cumplir con las reglas de integridad, estas reglas, junto con los dominios de los atributos, permiten mantener la estructura de la base de datos íntegra.

La primera de ellas es la regla de integridad de entidades, la cual indica que el la clave primaria no puede tener valor nulo, esto resulta coherente dado que la clave primaria debe identificar unívocamente a cada tupla, y el valor nulo no permite hacerlo.

La segunda regla es la regla de integridad referencial. Como explican [5] y [23], esta regla se aplica a las claves ajenas, e indica que si en alguna relación existe una clave ajena, dicha clave ajena debe coincidir con valores de la clave primaria a la que hace referencia, o en caso contrario debe tener un valor nulo. Esta regla sirve para averiguar que operaciones dejan a la base de datos en un estado inconsistente. Básicamente se aplica en 2 casos, actualizaciones y borrados. En cualquiera de esos dos casos se nos presentan una serie de opciones para devolver a la base de datos a un estado consistente. Podemos restringir los cambios, impidiendo que se realicen; o bien podemos actualizar en cascada, haciendo que dichos cambios se extiendan a toda la base de datos; o finalmente podemos establecer a la clave

ajena el valor nulo, devolviendo de esa forma la consistencia a la base de datos. Además de estas reglas, el usuario puede especificar las que considere oportunas para su aplicación aunque no todos los sistemas gestores de bases de datos permiten que se especifiquen nuevas reglas, o no verifican que se cumplan, quedando su implementación en manos del usuario.

La información contenida en una base de datos relacional, se maneja por medio de consultas, dichas consultas se efectúan mediante un lenguaje de manejo de datos, como puede ser el SQL, se trata de un lenguaje de cuarta generación es decir no procedural, el usuario decide que se va a hacer pero no como se hará. Esto simplifica mucho su uso y aumenta la productividad, dado que el usuario solo debe preocuparse por los datos que desea obtener.

Tras ver todo lo anterior, podemos concluir que una base de datos relacional es un conjunto de relaciones, llamadas tablas, estructuradas en tuplas (filas) y atributos (columnas), vinculadas entre sí por un campo en común y sobre las que se llevan a cabo una serie de operaciones para manipular los datos.

Referencias [1], [5], [6], [12], [13], [16] y [17]

1.2 Situación de Partida

Actualmente la ONG dispone de una base de datos en Access, a la cual se accede desde tres ordenadores diferentes con sistema operativo Windows. Debido al funcionamiento de Access, solo es posible el uso de la base de datos por parte de un usuario cada vez, lo cual resulta altamente ineficiente. Una consecuencia también de lo anterior es la imposibilidad de acceder de forma remota y segura.

La base de datos actual, es el resultado de la adición indiscriminada de campos y tablas a un modelo previo. A consecuencia de esto existe un gran número de campos vacios o redundantes en varias tablas y una seria dificultad para localizar la información relevante de manera ágil, todo ello conlleva que la base de datos ocupe más espacio del que realmente necesita, y al mismo tiempo hace que sea más lenta.

Debido a lo anterior cualquier pequeña modificación que se pretenda realizar en el modelo, resulta complicada con la estructura actual al encontrarse la información muy dispersa. Del mismo modo, si se desea modificar datos presentes en la base de datos, se hace necesario recorrer todas las tablas para verificar que no se produzca inconsistencia con los cambios realizados.

A lo anterior podemos añadir que dicho sistema no es multiplataforma, de forma que si en algún momento se desea realizar un cambio de sistema operativo, nos encontraremos con que necesitamos migrar la base de datos a otro sistema o seguir usando Windows, lo cual no es nada eficiente para una ONG por los costes de las licencias.

También es importante comentar el serio problema de seguridad que supone tener todos los datos en un solo archivo, en el caso de que un asaltante logre acceder al sistema operativo, bastaría con copiar el archivo de la base de datos para tener acceso a toda la información personal disponible en la misma.

Por este motivo la ONG desea tener una nueva base de datos que permita tener la información mejor organizada, con un acceso más sencillo a los datos, que permita a múltiples usuarios acceder de forma simultánea y haga posible el desarrollo de aplicaciones futuras de forma sencilla. En un principio se ha descartado el acceso online a la base de datos, pero es una funcionalidad que debe tenerse en cuenta en un futuro.

Otra de las características que deberá tener en cuenta la futura aplicación, es que los usuarios del sistema no disponen de conocimientos sobre la estructura de una base de datos, por lo cual será necesario un interfaz que encapsule el acceso a los datos, haciendo que el usuario tenga sólo opciones de manipulación de los datos, pero no pudiendo modificar la estructura de la

base de datos. De ser necesaria una ampliación de la estructura, para añadir más campos o tablas, sería necesario realizar dichos cambios modificando el diseño e implementando el código de nuevo. Esto limita las posibilidades de crecimiento de la base de datos, puesto que ahora no se podrán añadir campos en el instante a necesidad del usuario, pero a cambio la base de datos ganará en integridad de la información, limitando las posibilidades de que los datos se corrompan y se produzcan conflictos en la base de datos.

En la actualidad la obtención de los datos resulta lenta y en caso de necesitar datos estadísticos resulta muy compleja, se espera que al disponer de la información de forma más accesible y rápida, sea posible obtener datos estadísticos con mayor celeridad, de manera que puedan proporcionarse a instituciones científicas que requieran dicha información en un tiempo breve, lo que redundará en una mayor velocidad de las investigaciones.

1.3 Objetivos del proyecto

En este proyecto de fin de carrera se pretende diseñar, crear y establecer los procedimientos de gestión de una nueva base de datos, que usando los estándares web, permita optimizar el uso de la información por parte de la ONG.

- El principal objetivo de este proyecto es crear la estructura de la base de datos, esta estructura debe ser adaptable a futuros cambios y ampliaciones en función de las necesidades de la organización, también es necesario que la aplicación sea flexible, no se dispondrá siempre de todos los datos.

- Elección de las tecnologías y herramientas a utilizar para el desarrollo. Las tecnologías finalmente empleadas en el proyecto se han indicado en el apartado de Estado de la cuestión. Sobre todo se ha procurado que fueran de libre distribución y de uso mayoritario, de forma que sea fácilmente modificable en un futuro.

- Implementar un interfaz de acceso que permita a los usuarios interactuar de forma sencilla con la base de datos, de este modo permitiremos que un usuario no cualificado pueda introducir información en el sistema, siempre manteniendo la estructura de nuestra base de datos.

- Llevar a cabo las pruebas que permitan comprobar que la aplicación se encuentra en pleno funcionamiento.

- Aplicar los conocimientos adquiridos durante la carrera en el desarrollo de aplicaciones y bases de datos.

1.4 Organización del proyecto

El proyecto se estructura en los siguientes capítulos, cada apartado puede tener varios apartados:

Introducción

Descripción general del proyecto, las tecnologías que se van a utilizar y los objetivos que se pretenden conseguir con el mismo.

Estado de la cuestión

En este bloque se realiza un estudio de las diferentes partes del proyecto, como son las bases de datos. Se explican los conceptos básicos de manera que el lector tenga una idea básica de lo que se va a realizar en el proyecto.

Requisitos y diseño

En este apartado se indicaran de las especificaciones dadas por la ONG y todas aquellas decisiones de diseño que se hayan tomado durante el desarrollo del proyecto.

Implementación

En este bloque se desarrollará la implementación misma del proyecto, explicando todas aquellas partes del código vitales para la aplicación.

Pruebas y usabilidad

En este apartado se detallaran todas las pruebas que se han hecho a la aplicación, así como las mejoras en la usabilidad de la misma

Conclusiones y futuros desarrollos

En este apartado se analizan los resultados obtenidos por el proyecto y se comprueba hasta qué punto se han cumplido los objetivos del mismo. Asimismo se analizan aquellas posibles vías de continuación en el desarrollo, y las mejoras que pueden llevarse a cabo en la aplicación, así como aquellas nuevas funcionalidades que se puedan añadir en un futuro.

Bibliografía

Referencias de la documentación utilizada durante el desarrollo del proyecto.

Capítulo 2: Estado de la cuestión

2.1 Bases de Datos

2.1.1 Descripción

Una base de datos es un conjunto de datos relacionados entre sí, que son almacenados con el fin de usarlos o procesarlos posteriormente. Pueden contener muchos tipos de datos y se clasifican de muchas formas en función de su contenido o de la forma en que lo almacenan.

2.1.2 Historia

Como podemos ver en [1] y [16], los predecesores de los sistemas de bases de datos fueron los sistemas de ficheros, estos aun se usan hoy en día, pero en algún momento surgió la necesidad de hacer evolucionar el sistema para poder procesar mayor cantidad de información.

Los primeros sistemas de bases de datos surgen con las industrias que necesitaban manejar mucha información. En 1884 Herman Hollerit crea una máquina perforadora de tarjetas para el censo de Estados Unidos, tardó 2 años y medio en obtener resultados contra los 7 años necesarios para realizar el censo anterior.

Según algunas fuentes los sistemas de bases de datos actuales tienen sus orígenes en el proyecto Apolo en los años sesenta. Debido a la enorme cantidad de datos que requería el proyecto y a la falta de un sistema eficiente para gestionarlos, la empresa encargada del proyecto, NAA (North American Aviation), desarrolló un software con una estructura jerárquica, es decir de pequeñas piezas que se unen para formar una más grande sucesivamente, posteriormente NAA se unió a IBM para desarrollar el IMS (Information Management System). Este sistema funcionaba sobre dispositivos de cintas magnéticas (almacenamiento en serie), puesto que los discos magnéticos no aparecieron hasta algunos años mas tarde.

Posteriormente General Electric desarrolló el IDS (Integrated Data Store), un nuevo sistema de bases de datos en red, cuya finalidad era representar aquellas relaciones que no podían ser representadas correctamente en los sistemas jerárquicos.

Con el fin de establecer un estándar para las bases de datos, representantes del gobierno y las empresas de EEUU formaron un grupo denominado DBTG (Data Base Task Group), este grupo presento un informe final en 1971, y aunque no fue aceptado por el ANSI (American

National Standars Institute), varios sistemas se desarrollaron siguiendo la propuesta, los conocidos como sistemas de red.

Los sistemas jerárquico y de red forman la primera generación de SGBD, pero tienen varios inconvenientes:

- Requieren programas muy complejos para cualquier consulta sea simple o compleja.
- Hay muy poca independencia de datos.
- No tienen un fundamento teórico.

En 1970, en los laboratorios de investigación de IBM, Edgar Frank Codd escribió un artículo presentando el modelo relacional y al mismo tiempo describiendo los problemas de los modelos anteriores. A partir de ese momento se empiezan a desarrollar muchos sistemas relacionales, apareciendo los primeros a finales de los años setenta y principios de los ochenta.

Uno de los primeros sistemas fue el System R, de IBM, cuya finalidad era probar la funcionalidad del modelo relacional, posteriormente aparecieron dos grandes desarrollos:

- El lenguaje de consultas estructurado SQL, que se ha convertido en el lenguaje estándar de los sistemas relacionales.
- Los SGBD relacionales DB2 y SLQ/DS de IBM, y ORACLE de ORACLE Corporation.

Hoy en día, existen cientos de SGBD relacionales, tanto para microordenadores como para sistemas multiusuario, aunque muchos no son completamente fieles al modelo relacional.

Los SGBD relacionales constituyen la segunda generación de los SGBD. Sin embargo el modelo relacional también tiene algunos fallos, como su limitada capacidad de modelado de datos. Tratando de resolver dicho problema se presentó en 1976 el modelo entidad-relación, que se ha convertido en la técnica más utilizada en el diseño de bases de datos.

Como respuesta a la complejidad creciente de las aplicaciones que requieren bases de datos han surgido nuevos modelos, el modelo orientado a objetos y el relacional extendido, estos representan la tercera generación de los SGBD.

Los sistemas de bases de datos presentan numerosas ventajas que se pueden dividir en dos grupos: las que se deben a la integración de datos y las que se deben a la interfaz común que proporciona el SGBD.

Referencias [1] y [16]

2.1.3 Tipos de Bases de Datos

Los siguientes son los modelos de bases de datos más utilizados, o más conocidos.

- **Jerárquicas**: Estas bases de datos tienen estructura de árbol, con un nodo raíz del que salen todos los nodos hijos conocidos como hojas. Es una estructura muy útil cuando se maneja gran cantidad de información puesto que permite crear estructuras estables con un gran rendimiento. Resulta poco eficiente cuando hay redundancia de datos.

- **En red**: Es una variación del modelo jerárquico, en este modelo un nodo puede tener varios padres - algo que no permitía el modelo jerárquico -. Es un avance con respecto al modelo jerárquico al solucionar el problema de la redundancia de datos, pero la dificultad de administrar una base de datos de este estilo hace que los usuarios más frecuentes sean programadores.

- **Relacional**: Es el modelo más utilizado en la actualidad, permite una estructuración muy sencilla de la información y es muy fácil de usar. En el modelo relacional, cada tabla está relacionada con otra por medio de un campo idéntico en sus entradas.

- **Orientada a Objetos**: Es un modelo bastante reciente, que ha surgido debido a las necesidades de aplicaciones más complejas que no están restringidas por el tipo de datos o el lenguaje de consulta. En una base de datos orientada a objetos cada objeto se asocia a sus datos (o atributos), a una serie de mensajes a los que responde y una serie de métodos cada uno de los cuales es el código que implementa el mensaje y devuelve un valor.

- **Deductivas**: Este modelo se basa en reglas lógicas, de tal forma que de la información introducida por el usuario y siguiendo dichas reglas lógicas podamos extraer nueva información.

- **Distribuidas**: Como indica su nombre la información se encuentra distribuida entre distintas ubicaciones, pero debe ser accesible desde cualquier punto de la red como si la información estuviera centralizada.

2.1.4 Las Bases de Datos hoy

A día de hoy el modelo relacional es el más utilizado debido a su funcionalidad y su sencillez. En este modelo los datos se almacenan en tablas, que se dividen en líneas y estas a su vez en campos. A su vez las tablas se relacionan entre sí mediante campos idénticos, de este modo se forman estructuras de tablas que permiten una gran flexibilidad.

Esta estructura resulta especialmente útil en entornos web, ya que unido a otras tecnologías tiene un potencial prácticamente ilimitado, permitiendo que gran cantidad de usuarios generen información y al mismo tiempo que muchos otros usuarios obtengan aquella información que quieren de forma rápida y sencilla.

2.1.5 Ventajas por la integración de datos

Al hacer uso de bases de datos, obtenemos varias ventajas por la integración de los mismos en una estructura común.

Una de las ventajas es el control sobre la redundancia de datos; los sistemas de ficheros suelen almacenar varias copias de los mismos datos en ficheros distintos, haciendo que se desperdicie mucho espacio de almacenamiento, y provocando fallos de consistencia entre los datos. En los sistemas de bases de datos todos estos ficheros están integrados, por lo que no se almacenan varias copias de los mismos datos. Sin embargo, en una base de datos no se puede eliminar la redundancia completamente, ya que en ocasiones es necesaria para modelar las relaciones entre los datos, o bien es necesaria para mejorar las prestaciones del sistema.

Otra ventaja es la consistencia de los datos, al eliminar o reducir al mínimo las redundancias de datos disminuye en gran medida el riesgo de que haya inconsistencias. Si un dato está almacenado una sola vez, cualquier actualización se debe realizar sólo una vez, de forma que se encuentra disponible para todos los usuarios inmediatamente, lo que redunda en una mayor eficiencia. Si un dato está duplicado y el sistema conoce esta redundancia, el propio sistema puede encargarse de garantizar que todas las copias sean consistentes. Aunque desgraciadamente, no todos los SGBD actuales se encargan de mantener la consistencia de forma autónoma.

También es conveniente tener en cuenta que mediante las bases de datos se puede obtener más información sobre la misma cantidad de datos, al encontrarse todos los datos integrados, se puede extraer información adicional sobre los mismos. Del mismo modo, una base de datos permite que un grupo amplio de usuarios pueda compartir la información de forma simple, un fichero presenta una complejidad mayor para ser compartido.

Es muy importante recordar que el uso de las bases de datos facilita el mantenimiento de estándares, ya que gracias a la integración es más fácil respetar los estándares establecidos a nivel de la empresa así como los nacionales e internacionales. Estos estándares pueden establecerse sobre el formato de los datos para facilitar su intercambio, pueden ser estándares de documentación, procedimientos de actualización y también reglas de acceso.

En conclusión la integración de datos permite una gestión más eficiente de los mismos, reduciendo costes y ampliando la funcionalidad de los sistemas.

2.1.6 Ventajas por la existencia del SGBD

Como indica [5], al existir un SGBD se obtienen ventajas que no solo provienen de la integración de los datos. La integridad de la base de datos se refiere a la validez y la consistencia de los datos almacenados. Normalmente, la integridad se expresa mediante restricciones o reglas que no se pueden violar. Estas restricciones se pueden aplicar tanto a los datos, como a sus relaciones, y es el SGBD quien se debe encargar de mantenerlas.

Pero el SGBD también proporciona mejoras en la seguridad, se trata de la protección de la base de datos frente a usuarios no autorizados, y sin unas buenas medidas de seguridad, la integración de datos en los sistemas de bases de datos hace que éstos sean más vulnerables que en los sistemas de ficheros. Sin embargo, los SGBD permiten mantener la seguridad mediante el establecimiento de claves para identificar al personal autorizado a utilizar la base de datos. Las autorizaciones se pueden realizar a nivel de operaciones, de modo que un usuario puede estar autorizado a consultar ciertos datos pero no a actualizarlos, por ejemplo.

También es importante no perder la accesibilidad de los datos para ganar seguridad. Muchos SGBD proporcionan lenguajes de consultas o generadores de informes que permiten al usuario hacer cualquier tipo de consulta sobre los datos, sin que sea necesario que un programador escriba una aplicación que realice esa tarea.

Un SGBD proporciona muchas de las funciones estándar que el programador necesita escribir en un sistema de ficheros, dispone de todas las rutinas de manejo de ficheros típicas de los programas de aplicación. El hecho de disponer de estas funciones permite al programador centrarse mejor en la función específica requerida por los usuarios, sin tener que preocuparse de los detalles de implementación de bajo nivel. Muchos SGBD también proporcionan un entorno consistente en un conjunto de herramientas que simplifican, en gran medida, el desarrollo de las aplicaciones que acceden a la base de datos. Gracias a estas herramientas, el programador puede ofrecer una mayor productividad en un tiempo menor.

Otra ventaja de los SGBD es la facilidad de mantenimiento. En los sistemas de ficheros, las descripciones de los datos se encuentran inmersas en los programas de aplicación que los manejan. Esto hace que los programas sean dependientes de los datos, de modo que un cambio en su estructura, o un cambio en el modo en que se almacena en disco, requiere cambios importantes en los programas cuyos datos se ven afectados. Sin embargo, los SGBD separan las descripciones de los datos de las aplicaciones. Esto es lo que se conoce como independencia de datos, gracias a la cual se simplifica el mantenimiento de las aplicaciones que acceden a la base de datos.

Los SGBD permiten aumentar la concurrencia, es decir, coincidir a varios usuarios sobre la misma información. En algunos sistemas de ficheros, si hay varios usuarios accediendo simultáneamente a un mismo fichero, es posible que se produzcan conflictos entre ellos de modo que se pierda información o, incluso, que se pierda la integridad. La mayoría de los SGBD gestionan el acceso a la base de datos y garantizan que no ocurran problemas de este tipo.

Para terminar, es importante la facilidad que permiten los SGBD en los servicios de copias de seguridad y de recuperación ante fallos. Muchos sistemas de ficheros dejan que sea el usuario quien proporcione las medidas necesarias para proteger los datos ante fallos en el sistema o en las aplicaciones, haciendo copias de seguridad cada día, y si se produce algún fallo, utilizando estas copias para restaurarlos. En este caso, todo el trabajo realizado sobre los datos desde que se hizo la última copia de seguridad se pierde y se tiene que volver a realizar. Sin embargo, los SGBD actuales funcionan de modo que se minimiza la cantidad de trabajo perdido cuando se produce un fallo.

2.1.7 Inconvenientes de los sistemas de bases de datos

Lógicamente los SGBD no son perfectos y tal como viene en [5], tienen una serie de inconvenientes, que es necesario valorar antes de implementarlos en nuestro sistema.

Los SGBD son conjuntos de programas muy complejos, es preciso comprender muy bien su funcionalidad para poder sacar un buen partido de ellos. Asimismo requieren una gran cantidad de espacio en disco y de memoria para funcionar de forma eficiente.

El coste económico de un SGBD es muy variable, existen algunos gratuitos y otros que para dar servicio a cientos de usuarios pueden tener costes elevados. En muchos casos tienen una cuota de mantenimiento anual.

Tanto el SGBD, como la propia base de datos, pueden hacer que sea necesario adquirir más espacio de almacenamiento. Además, para alcanzar las prestaciones deseadas, es posible que sea necesario adquirir un ordenador más grande o que se dedique solamente al SGBD. Todo esto hará que la implantación de un sistema de bases de datos sea más cara.

En algunas ocasiones, el coste del SGBD y del equipo informático que sea necesario adquirir para su buen funcionamiento, es insignificante comparado con el coste de convertir la aplicación actual en un sistema de bases de datos. Este coste incluye el de enseñar a la plantilla a utilizar estos sistemas y, probablemente, el coste del personal especializado para ayudar a realizar la conversión y poner en marcha el sistema. Esta es una de las razones principales por las que algunas empresas y organizaciones se resisten a cambiar su sistema actual de ficheros por un sistema de bases de datos.

Una ventaja de los sistemas de ficheros respecto a las bases de datos, es que el primero está escrito para una aplicación específica, por lo que sus prestaciones suelen ser muy buenas. Sin embargo, los SGBD están escritos para ser más generales y ser útiles en muchas aplicaciones, lo que puede hacer que algunas de ellas no sean tan rápidas como antes.

También se debe tener en cuenta, el hecho de que al encontrarse todo centralizado en el SGBD, el sistema resulta más vulnerable ante los fallos que puedan producirse.

2.1.8 Usuarios de la base de datos

Existen varios tipos de usuarios en función de sus permisos y su nivel de acceso a la base de datos:

- **Administrador**: Se encarga del diseño físico de la base de datos y de su implementación, realiza el control de la seguridad y de la concurrencia, asimismo mantiene el sistema para que se encuentre siempre operativo. Debe conocer muy bien el SGBD y el equipo informático sobre el que funciona.

- **Diseñador**: Realiza el diseño lógico de la base de datos, debe identificar los datos, las relaciones entre esos datos. Es necesario que el diseñador tenga un conocimiento amplio de la empresa y debe implicar desde el principio a los usuarios finales para obtener un mejor resultado.

- **Programador**: Persona que implementa las transacciones e interfaces que usaran los usuarios finales. Su trabajo será el que permita a dichos usuarios interactuar con los datos.

- **Usuario**: Consulta y edita los datos de la base de datos mediante un lenguaje de consultas de alto nivel, haciendo uso del interfaz realizado por el programador. Dado que no necesita tener conocimientos específicos debe evitarse que tenga posibilidad de realizar cambios críticos.

Referencias [5], [6], [12], [13] y [17]

2.2 MySQL

2.2.1 Descripción

MySQL es el Sistema Gestor de Bases de Datos de código abierto más popular en Internet. Está diseñado para entornos de producción con mucha carga de trabajo, así como para integrarse en software para ser distribuido. MySQL es una marca registrada de MySQL AB. Al ser un servidor de bases de datos relacionales, MySQL se convierte en una herramienta veloz en la accesibilidad a los datos introducidos en las distintas tablas independientes que forman las bases de datos de este lenguaje.

MySQL Server fue diseñado originalmente para trabajar con bases de datos de tamaño medio (de 10 a 100 millones de registros, unos 100MB por tabla) en máquinas pequeñas. Hoy en día ya soporta bases de datos de gran tamaño con terabytes de información, pero el código todavía puede compilarse en una versión reducida.

2.2.2 Ventajas

El servidor de bases de datos MySQL es muy rápido, seguro, y fácil de instalar y usar.

Ha sido desarrollado para manejar bases de datos mucho más rápido que las soluciones existentes y ha estado siendo usado con éxito en ambientes de producción muy exigentes por varios años. Aunque se encuentra en desarrollo constante, ofrece hoy en día, un amplio conjunto de funciones.

Su velocidad y seguridad hacen que sea un servidor bastante apropiado para acceder a bases de datos en Internet. Asimismo se trata de un sistema libre que no requiere pago por licencia.

MySQL se relaciona con el estándar ANSI/ISO SQL y trata de cumplirlas en lo más posible sin disminuir la velocidad del software y su usabilidad. Este estándar SQL ha ido evolucionando desde 1986.

Soporta bases de datos transaccionales y no transaccionales para satisfacer uso crítico y uso pesado en entornos Web.

2.2.2.1 Características principales

Ha sido escrito en C y en C++, y dispone de APIs (Application Programming Interface) para C, C++, Eiffel, Java, Perl, PHP, Python, Ruby y Tcl. Asimismo está probado en un amplio número de compiladores distintos.

Se trata de un sistema multiplataforma. Aunque las plataformas en las que MySQL obtiene un mejor rendimiento en este momento son x86 con SuSE Linux, y ReiserFS, o distribución linux similar, y SPARC con Solaris.

Dispone de un sistema de reserva de memoria basado en subprocesos (threads) que permite un uso sencillo de múltiples CPUs si están disponibles.

Proporciona sistemas de almacenamiento, transaccionales y no transaccionales, de forma que podamos usar el más conveniente a nuestra aplicación. Un sistema transaccional es aquel que mantiene la integridad de los datos, de forma que en caso de fallo durante la transacción, se deshace todo lo realizado para volver al punto anterior a la misma. También usa tablas en disco B-tree (MyISAM) muy rápidas con compresión de índice, esto hace que las consultas sean más rápidas y eficientes. Además podemos realizar uniones (joins) muy rápidos usando una unión múltiple de un paso optimizada.

Permite añadir de una forma sencilla otro sistema de almacenamiento. Esto es útil si desea añadir una interfaz SQL para una base de datos propia. Las funciones SQL están implementadas usando una librería muy optimizada y deben ser tan rápidas como sea posible. Normalmente no hay reserva de memoria tras toda la inicialización para consultas. Dispone de tablas hash en memoria, que son usadas como tablas temporales.

El servidor está disponible como un programa separado para usar en un entorno de red cliente/servidor. También está disponible como biblioteca y puede ser incrustado en aplicaciones autónomas. Dichas aplicaciones pueden usarse por sí mismas o en entornos donde no hay red disponible.

2.2.2.2 Tipos de columnas

MySQL dispone de diversos tipos de columnas, de forma que el usuario pueda disponer de la más conveniente a sus necesidades:

Tenemos los tipos numéricos, que engloban enteros con y sin signo, de 1, 2, 3, 4 y 8 bytes de longitud, son TINYINT, SMALLINT, MEDIUMINT, INT, BIGINT, también disponemos de tipos decimales DECIMAL, NUMERIC, FLOAT, REAL, y DOUBLE PRECISION.
Por otro lado están los tipos de texto CHAR, VARCHAR, BINARY, VARBINARY, BLOB, TEXT, ENUM, y SET. Permiten guardar tipos de texto de tamaño variable y con formatos distintos.

Asimismo MySQL dispone de varios formatos de fecha y hora, que son DATE, TIME, DATETIME, TIMESTAMP, YEAR. Cada uno de ellos almacena el tiempo de forma distinta, desde la fecha con la hora, minutos y segundos, hasta simplemente el año.

2.2.2.3 Sentencias y funciones

MySQL intenta cumplir al máximo con las especificaciones del estándar SQL, pese a ello no siempre es posible dar un soporte completo del mismo. A continuación se indican algunas de las principales características del estándar que disponen de soporte en MySQL

- Soporte completo para operadores y funciones en las cláusulas de consultas SELECT y WHERE.

- Soporte completo para las cláusulas SQL GROUP BY y ORDER BY.

- Soporte de funciones de agrupación (COUNT(), COUNT(DISTINCT ...), AVG(), STD(), SUM(), MAX(), MIN(), y GROUP_CONCAT()).

- Soporte para LEFT OUTER JOIN y RIGHT OUTER JOIN cumpliendo estándares de sintaxis SQL y ODBC.

- Soporte para alias en tablas y columnas como lo requiere el estándar SQL.

- DELETE, INSERT, REPLACE, y UPDATE devuelven el número de filas que han cambiado (han sido afectadas). Es posible devolver el número de filas que serían afectadas usando un flag al conectar con el servidor.

- El comando específico de MySQL SHOW puede usarse para obtener información acerca de la base de datos, el motor de base de datos, tablas e índices.

- El comando EXPLAIN puede usarse para determinar cómo el optimizador resuelve una consulta.

- Los nombres de funciones no colisionan con los nombres de tabla o columna.

- Puede mezclar tablas de distintas bases de datos en la misma consulta.

2.2.2.4 Seguridad

MySQL utiliza seguridad basada en Listas de Control de Acceso (ACLs). También hay algún soporte para conexiones cifradas mediante SSL entre clientes y servidores MySQL.

Para mayor seguridad es importante seguir las siguientes recomendaciones:

Solo la cuenta root de MySQL debe tener acceso a la tabla user en la base de datos mysql. La clave cifrada es la verdadera clave en MySQL**.** Cualquiera que sepa cuál es la clave que hay en la tabla user y tenga acceso a la máquina servidor de la cuenta registrada puede acceder fácilmente como ese usuario.

Estudie el sistema de privilegios de acceso de MySQL. Las sentencias GRANT y REVOKE se utilizan para controlar el acceso a MySQL. No otorgue más privilegios de los necesarios. Nunca otorgue privilegios a un mismo usuario sin tener en cuenta el equipo desde el que se conecta.

Comprobaciones:

- Pruebe el comando mysql -u root. Si es capaz de conectar al servidor sin la necesidad de introducir una clave, tiene problemas.

- Utilice la sentencia SHOW GRANTS y compruebe quién tiene acceso a qué. Después utilice la sentencia REVOKE para denegar los privilegios que no son necesarios.

No almacene ninguna clave sin cifrar en su base de datos. Si alguien tuviera acceso a su ordenador, el intruso podría obtener la lista completa de claves y utilizarlas. En vez de eso, utilice MD5(), SHA1(), o cualquier otra función de hashing de un sentido.

No elija claves que puedan aparecer en un diccionario. Existen programas especiales para romperlas. Incluso claves como ``xperro98'' son muy malas. Es mucho mejor ``oweei98'', que contiene la misma palabra ``perro'' pero escrita desplazándose una tecla a la izquierda en un teclado QWERTY convencional. Otro método es usar ``Mtupc'', que ha sido tomada de las primeras letras de cada palabra de la frase ``María tuvo un pequeño corderito. ‘‘Así es fácil de recordar y escribir, pero difícil de adivinar para cualquiera que no la conozca.

Invierta en un firewall. Le protegerá de al menos el 50% de todos los tipos de vulnerabilidades de cualquier software. Ponga MySQL tras el firewall o en una zona desmilitarizada (DMZ).

Comprobaciones:

- Intente escanear sus puertos desde Internet utilizando una herramienta como nmap. MySQL utiliza el puerto 3306 por defecto. Este puerto no debería ser accesible desde lugares no confiables. Otra manera simple de probar si el puerto MySQL está abierto o no es intentar el siguiente comando desde alguna máquina remota, donde server_host es la máquina en la que su servidor MySQL se está ejecutando: shell> telnet server_host 3306

- Si consigue conectar y algunos caracteres extraños, el puerto está abierto, y debería cerrarlo en su firewall o router, a menos que tenga una buena razón para mantenerlo abierto. Si el comando telnet no consigue conectar o la conexión es rechazada, entonces el puerto se encuentra bloqueado, que es como queremos que esté.

No confíe en ningún dato enviado por los usuarios de sus aplicaciones. Pueden intentar engañar a su código introduciendo secuencias de caracteres especiales en formularios webs, URLs, o cualquier aplicación que haya desarrollado. Asegúrese de que su aplicación permanece segura si un usuario introduce algo como ``; DROP DATABASE mysql; ‘‘. Este es un ejemplo algo extremo, pero los mayores agujeros de seguridad y pérdidas de datos pueden ocurrir como resultado de hackers utilizando técnicas similares, si no se está preparado para ellas.

Un error común es proteger únicamente valores de tipo cadena de caracteres. Recuerde comprobar los datos numéricos también. Si una aplicación genera una consulta como `“SELECT * FROM table WHERE ID=123”` cuando un usuario introduce el valor 123, el usuario podría introducir el valor `“123 OR 1=1”` para provocar que la aplicación genere la consulta `“SELECT * FROM table WHERE ID=123 OR 1=1”`.

Como resultado, el servidor extraerá todos los registros en la tabla. Esto, además de exponer cada registro, causa una carga excesiva en el servidor. La manera más simple de protegerse frente a este tipo de ataque es utilizar comillas simples alrededor de las constantes numéricas: `“SELECT * FROM table WHERE ID='234'”`. Si el usuario entrase información extra, todo sería parte de la cadena de caracteres. En un contexto numérico, MySQL automáticamente convierte esta cadena en un número, y elimina cualquier carácter no numérico del final que la cadena pueda contener.

A veces la gente piensa que si una base de datos contiene sólo datos de dominio público, no tiene por qué ser protegida. Esto es incorrecto. Aunque sea admisible mostrar cualquier

registro de la base de datos, siempre se debería proteger contra ataques de tipo denegación de servicio (por ejemplo, aquellos que se basan en la técnica del párrafo precedente, que causan que el servidor malgaste recursos). Si no, el servidor podría quedar inservible para sus usuarios legítimos.

Comprobaciones:

- Intente introducir comillas simples y dobles (''' y '"') en todos sus formularios web. Si obtiene cualquier clase de error MySQL, investigue el problema sin demora.

- Intente modificar las URLs dinámicas añadiendo las cadenas %22 ('"'), %23 ('#'), y %27 (''').

- Intente modificar los tipos de datos en las URLs dinámicas de tipos numéricos a alfanuméricos, usando los caracteres mostrados en los ejemplos previos. Su aplicación debería ser segura contra estos y otros ataques similares.

- Intente introducir letras, espacios, y símbolos especiales en vez de números en los campos numéricos. Su aplicación debería eliminarlos antes de pasarlos a MySQL, o en todo caso generar un error.

- Compruebe el tamaño de los datos antes de pasárselos a MySQL.

- Haga que su aplicación se conecte a la base de datos utilizando un nombre de usuario diferente del que utiliza para tareas administrativas. No dé a sus aplicaciones ningún acceso que no necesiten.

No transmita datos sin cifrar por Internet. Esta información es accesible para cualquiera que tenga el tiempo y la habilidad para interceptarla y utilizarla para sus propios propósitos. En vez de eso, utilice un protocolo de cifrado como SSL o SSH. MySQL soporta conexiones SSL internas desde la versión 4.0.0. El redireccionamiento de puertos de SSH se puede utilizar para crear un túnel cifrado (y comprimido) para la comunicación.

Si se siguen todas estas recomendaciones el sistema se encontrará muy seguro, pero nunca se debe bajar la guardia.

2.2.2.5 Escalabilidad y limite

Soporte a grandes bases de datos. Usamos MySQL Server con bases de datos que contienen 50 millones de registros. También conocemos usuarios que usan MySQL Server con 60.000 tablas y acerca de 5.000.000 de registros.

Se permiten hasta 64 índices por tabla (32 antes de MySQL 4.1.2). Cada índice puede consistir desde 1 hasta 16 columnas o partes de columnas. El máximo ancho de límite son 1000 bytes (500 antes de MySQL 4.1.2).Un índice puede usar prefijos de una columna para los tipos de columna CHAR, VARCHAR, BLOB, o TEXT.

2.2.2.6 Conectividad

Los clientes pueden conectar con el servidor MySQL usando sockets TCP/IP en cualquier plataforma. En sistemas Windows de la familia NT (NT,2000,XP, o 2003), los clientes pueden usar named pipes para la conexión. En sistemas Unix, los clientes pueden conectar usando ficheros socket Unix.
En MySQL 5.0, los servidores Windows soportan conexiones con memoria compartida si se inicializan con la opción --shared-memory. Los clientes pueden conectar a través de memoria compartida usando la opción --protocol=memory.

La interfaz para el conector ODBC (MyODBC) proporciona a MySQL soporte para programas clientes que usen conexiones ODBC (Open Database Connectivity). Por ejemplo, puede usar MS Access para conectar al servidor MySQL. Los clientes pueden ejecutarse en Windows o Unix. El código fuente de MyODBC está disponible. Todas las funciones para ODBC 2.5 están soportadas, así como muchas otras. Consulte Sección 25.1, “MySQL Connector/ODBC”.

La interfaz para el conector J MySQL proporciona soporte para clientes Java que usen conexiones JDBC. Estos clientes pueden ejecutarse en Windows o Unix. El código fuente para el conector J está disponible. Consulte Sección 25.4, “MySQL Connector/J”.

2.2.2.7 Localización

El servidor puede proporcionar mensajes de error en casi cualquier idioma. Asimismo soporta diversos conjuntos de caracteres, entre los cuales se incluyen latin1 (ISO-8859-1), german, big5, ujis, y más. Por ejemplo, los caracteres escandinavos 'â', 'ä' y 'ö' están permitidos en nombres de tablas y columnas. También se encuentra disponible el soporte para Unicode.

Todos los datos se guardan en el conjunto de caracteres elegido, las comparaciones para columnas normales de cadenas de caracteres no son sensibles a mayúsculas.

La ordenación se realiza de acuerdo con el conjunto de caracteres que se haya elegido. Es posible cambiarla cuando arranca el servidor. MySQL Server soporta diferentes conjuntos de caracteres que deben ser especificados en tiempo de compilación y de ejecución.

2.2.2.8 Clientes y Herramientas

MySQL server tiene soporte para comandos SQL para chequear, optimizar, y reparar tablas. Estos comandos están disponibles a través de la línea de comandos y el cliente mysqlcheck. MySQL también incluye myisamchk, una utilidad de línea de comandos muy rápida para efectuar estas operaciones en tablas MyISAM. Consulte Capítulo 5, Administración de bases de datos.

Todos los programas MySQL pueden invocarse con las opciones --help o -? para obtener asistencia en línea.

2.2.2.9 Dimensiones máximas de las tablas

La siguiente tabla lista algunos ejemplos de límites de tamaño de ficheros de sistemas operativos.

Sistema operativo	Tamaño máximo de fichero
Linux 2.2-Intel 32-bit	2GB (LFS: 4GB)
Linux 2.4 (usando sistema de ficheros ext3)	4TB
Solaris 9/10	16TB
Sistema de ficheros NetWare w/NSS	8TB
win32 w/ FAT/FAT32	2GB/4GB
win32 w/ NTFS	2TB (posiblemente mayor)
MacOS X w/ HFS+	2TB

Referencias [2], [8], [11] y [17]

2.3 PHP

2.3.1 Descripción

PHP ("**P**HP **H**ypertext **P**re-processor") es un lenguaje de script, diseñado para, entre otras cosas, incrementar el dinamismo de las páginas web. Originalmente se trataba de un conjunto de macros concebidas para ayudar en el mantenimiento de páginas web. Desde entonces, sus características han ido creciendo hasta convertirse en un lenguaje de programación completo, capaz de manejar entornos que integran grandes bases de datos.

PHP se desarrolla en dos versiones paralelas: Para la versión 4, su última publicación es 4.4.9, y 5.2.9 para la versión 5.
En las referencias [2], [7], [10] y [15] podemos encontrar mas información relevante sobre este tema.

2.3.2 Ventajas

Es un lenguaje de programación muy completo, estructurado, permite manejar con facilidad bases de datos y se adapta muy bien al entorno web. Asimismo su sintaxis es similar a la del lenguaje de programación C, es muy rápido y simple, lo que lo convierte en uno de los lenguajes más populares en la red.

Además PHP es un desarrollo "Open Source". Es decir, es gratuito, puedes ver y modificar el código fuente de la aplicación siempre y cuando cumplas con su licencia PHP.

Es un lenguaje de script utilizado en los servidores web y es allí donde se ejecuta. Es por eso que una página que incluya código PHP será interpretada en el servidor antes de mandarla al cliente (en este caso, un usuario informático que haya pedido una página web a través de cualquier navegador convencional). La página finalmente enviada ya no incluye el código PHP, solo el HTML típico de toda web.

Las páginas que contienen código PHP cambian antes de que el usuario las vea, dependiendo de ciertas condiciones. Esto se puede utilizar, por ejemplo, para escribir algo en esa página, crear una tabla con el mismo número de filas como veces ha entrado el usuario en ese sitio o integrar en la página una base de datos como MySQL. Es decir, PHP convierte una página estática en otra dinámica.

Debido al amplio grupo de usuarios que utilizan PHP, existen gran cantidad de Frameworks que facilitan el desarrollo de aplicaciones.

2.3.3 Historia

2.3.3.1 PHP/FI

PHP es el heredero de un producto anterior, llamado PHP/FI. Este último fue creado por Rasmus Lerdorf en 1995, inicialmente como un simple conjunto de scripts de Perl para controlar los accesos a su trabajo online. Llamó a ese conjunto de scripts 'Personal Home Page Tools'. Conforme requería más funcionalidad, Rasmus fue escribiendo una implementación C mucho mayor, capaz de comunicarse con bases de datos, y que permitía a los usuarios desarrollar sencillas aplicaciones Web dinámicas. Rasmus eligió liberar el código fuente de PHP/FI para que cualquiera pudiese utilizarlo, además de arreglar errores y mejorar el código.

PHP/FI, que se mantuvo para páginas personales y como intérprete de formularios, incluía algunas de las funcionalidades básicas de PHP tal y como lo conocemos hoy. Tenía variables como las de Perl, interpretación automática de variables de formulario y sintaxis embebida HTML. La sintaxis por sí misma era similar a la de Perl, aunque mucho más limitada, simple y algo inconsistente.

Por 1997, PHP/FI 2.0, la segunda escritura de la implementación en C, tuvo un seguimiento estimado de varios miles de usuarios en todo el mundo, con aproximadamente 50.000 dominios informando que lo tenían instalado, sumando alrededor del 1% de los dominios de Internet.

PHP/FI 2.0 no se liberó oficialmente hasta Noviembre de 1997, después de gastar la mayoría de su vida en desarrollos beta. Fue sucedido en breve tiempo por las primeras versiones alfa de PHP 3.0.

2.3.3.2 PHP 3

PHP 3.0 era la primera versión que se parecía fielmente al PHP tal y como lo conocemos hoy en día. Fue creado por Andi Gutmans y Zeev Zuraski en 1997 reescribiéndolo completamente, después de que encontraran que PHP/FI 2.0 tenía pocas posibilidades para desarrollar una aplicación comercial que estaban desarrollando para un proyecto universitario.

En un esfuerzo para cooperar y empezar a construir sobre la base de usuarios de PHP/FI existente, Andi, Rasmus y Zeev decidieron cooperar y anunciar PHP 3.0 como el sucesor oficial de PHP/FI 2.0, interrumpiéndose en su mayor parte el desarrollo de PHP/FI 2.0.

Una de las mejores características de PHP 3.0 era su gran extensibilidad. Además de proveer a los usuarios finales de una sólida infraestructura para muchísimas bases de datos, protocolos y APIs, las características de extensibilidad de PHP 3.0 atrajeron a docenas de desarrolladores a unirse y enviar nuevos módulos de extensión. Sin duda, ésta fue la clave del enorme éxito de PHP 3.0. Otras características clave introducidas en PHP 3.0 fueron el soporte de sintaxis orientado a objetos y una sintaxis de lenguaje mucho más potente y consistente.

Todo el nuevo lenguaje fue liberado bajo un nuevo nombre, que borraba la implicación de uso personal limitado que tenía el nombre PHP/FI 2.0. Se llamó 'PHP' a secas, con el significado de ser un acrónimo recursivo - PHP: Hypertext Preprocessor.

A finales de 1998, PHP creció hasta una base de instalación estimada de decenas de millares de usuarios y cientos de miles de sitios Web informando de su instalación. En su apogeo, PHP 3.0 estaba instalado en aproximadamente un 10% de los servidores Web en Internet.
PHP 3.0 se liberó oficialmente en Junio de 1998, después de haber gastado unos 9 meses en pruebas públicas.

2.3.3.3 PHP 4

En el invierno de 1998, poco después del lanzamiento oficial de PHP 3.0, Andi Gutmans y Zeev Suraski comenzaron a trabajar en la reescritura del núcleo de PHP. Los objetivos de diseño fueron mejorar la ejecución de aplicaciones complejas, y mejorar la modularidad del código base de PHP. Estas aplicaciones se hicieron posibles por las nuevas características de PHP 3.0 y el apoyo de una gran variedad de bases de datos y APIs de terceros, pero PHP 3.0 no fue diseñado para el mantenimiento tan complejo de aplicaciones eficientemente.

El nuevo motor, apodado 'Motor Zend' (comprimido de Zeev y Andi), alcanzó estos objetivos de diseño satisfactoriamente, y se introdujo por primera vez a mediados de 1999. PHP 4.0, basado en este motor, y acoplado con un gran rango de nuevas características adicionales, fue oficialmente liberado en Mayo de 2000, casi dos años después que su predecesor, PHP 3.0.

Además de la mejora de ejecución de esta versión, PHP 4.0 incluía otras características clave como el soporte para la mayoría de los servidores Web, sesiones HTTP, buffers de salida, formas más seguras de controlar las entradas de usuario y muchas nuevas construcciones de lenguaje.

Hoy, se estima que PHP es usado por cientos de miles de programadores y muchos millones de sitios informan que lo tienen instalado, sumando más del 20% de los dominios en Internet.

El equipo de desarrollo de PHP incluye docenas de programadores, así como otras docenas de personas trabajando en proyectos relacionados con PHP como PEAR y el proyecto de documentación.

El 7 de agosto de 2008 se publicó la última versión estable de PHP 4 que en estos momentos ya no tiene soporte dado que los esfuerzos de los desarrolladores se han centrado en PHP 5

2.3.3.4 PHP 5

En 2004, salió por fin la nueva versión de PHP, la versión 5. Haciendo uso de una nueva versión del motor Zend, la 2.0.

Entre las muchas mejoras de PHP 5 se encuentra un soporte mejorado de la orientación a objetos, un mayor rendimiento y mejor optimización del uso de memoria. Su implantación ha sido lenta debido al enorme arraigo de PHP 4 y a algunos problemas de funcionamiento que se han producido en sus primeras versiones.

Hoy en día aun coexisten aplicaciones en PHP 4 y PHP 5, ya que muchos servicios no han migrado por los conflictos que existen entre ambas versiones. En cualquier caso se considera que PHP 5 es un paso intermedio hasta la salida de PHP 6 que incluirá un mayor soporte para la orientación a objetos y soporte pleno para UNICODE.

2.3.4 Características

PHP soporta una gran cantidad de bases de datos: MysSQL, PostgreSQL, Oracle, MS SQL Server, Sybase mSQL, Informix, etc. También dispone de integración con varias bibliotecas externas, que permiten generar desde documentos PDF hasta analizar código XML.

Se trata de un lenguaje fácil de mantener y de poner al día, y que ofrece una solución simple para el desarrollo de páginas Web. Al ser usado por un gran número de programadores y ser un producto de código abierto, los errores son subsanados rápidamente y es factible encontrar ayuda con cualquier problema.

Cada día surgen nuevas mejoras y extensiones que permiten ampliar las funcionalidades del lenguaje. Cualquier cosa que se pueda realizar con un script CGI, como procesar información en formularios, foros de discusión, etc. puede ser realizado mediante PHP.

En sus últimas versiones, PHP dispone de soporte para Orientación a Objetos, haciendo aún más completo el lenguaje y ampliando sus posibilidades.

2.3.5 Referencias del lenguaje

2.3.5.1 Sintaxis básica

Para interpretar un archivo, php simplemente interpreta el texto del archivo hasta que encuentra uno de los caracteres especiales que delimitan el inicio de código PHP. El intérprete ejecuta entonces todo el código que encuentra, hasta que encuentra una etiqueta de fin de código, que le dice al intérprete que siga ignorando el código siguiente. Este mecanismo permite embeber código PHP dentro de HTML: todo lo que está fuera de las etiquetas PHP se deja tal como está, mientras que el resto se interpreta como código.

Hay cuatro conjuntos de etiquetas que pueden ser usadas para denotar bloques de código PHP. De estas cuatro, sólo 2 (<?php. . .?> y <script language="php">. . .</script>) están siempre disponibles; el resto pueden ser configuradas en el fichero de php.ini para ser o no aceptadas por el intérprete. Si se pretende embeber código PHP en XML o XHTML, será obligatorio el uso del formato <?php. . .?> para la compatibilidad con XML.

2.3.6 Frameworks

Se trata de estructuras de soporte que proporcionan una serie de librerías y protocolos para la programación de aplicaciones. Permiten que los programadores se centren en las partes propias de la aplicación proporcionando soluciones ya implementadas para los problemas más comunes.

En PHP existe una gran variedad de frameworks que permiten entre otras muchas cosas encapsular el acceso a las bases de datos. A continuación se indican algunos de los frameworks de PHP más utilizados, no son todos ya que fácilmente superan la veintena, así que este análisis se va a centrar en 3 de los más conocidos.

2.3.6.1 CakePHP

Es un marco de desarrollo de código abierto para PHP, dispone de una amplia comunidad de usuarios, es compatible con PHP4 y PHP5, está basado en el modelo vista-controlador y dispone de validación integrada y ayudantes para el desarrollo de AJAX, HTML, formularios, etc.

Dispone de abundante documentación y permite un desarrollo más rápido de las aplicaciones.

Referencias [3] y [4]

2.3.6.2 Zend

Se trata de un Framework de uso profesional, soporta únicamente PHP5 y también dispone de una amplia comunidad de usuarios, así como de amplia documentación. Dispone de las mismas ventajas que CakePHP, además de disponer de mayor soporte y una oferta formativa mayor, aunque se trata de un Framework de uso más complejo y el uso de algunas de sus ventajas no son de libre acceso.

Referencia [18]

2.3.6.3 Symfony

También se trata de un Framework profesional con una comunidad de usuarios muy amplia, también ha sido implementado para PHP5 y tiene ventajas similares a los frameworks anteriores, aunque su rendimiento es algo más lento. También dispone de soporte y oferta formativa más amplia que CakePHP.

Referencia [14]

Referencias del apartado 2.3: [2], [3], [4], [7], [9], [10], [14], [15], [17] y [18]

2.4 HTML

2.4.1 Descripción

Para publicar información y distribuirla globalmente, se necesita un lenguaje común, una especie de lengua franca que todos los ordenadores puedan comprender. El lenguaje usado por la World Wide Web es el HTML ("**H**yper**T**ext **M**arkup **L**anguage") un lenguaje de marcado que se usa para estructurar textos, en un principio su finalidad era describir documentos científicos, pero sucesivas adaptaciones lo han convertido en apto para todo tipo de documentos.

El HTML da a los autores las herramientas para:

- Publicar documentos en línea con encabezados, textos, tablas, listas, fotos, etc.
- Obtener información en línea a través de vínculos de hipertexto, haciendo clic con el ratón.
- Diseñar formularios para realizar transacciones con servicios remotos, para buscar información, hacer reservas, pedir productos, etc.

Incluir hojas de cálculo, videoclips, sonidos, y otras aplicaciones directamente en sus documentos.

2.4.2 Historia

El HTML fue desarrollado originalmente por Tim Berners-Lee mientras estaba en el CERN, y fue popularizado por el navegador Mosaic desarrollado en el NCSA. Durante los años 90 ha proliferado con el crecimiento explosivo de la Web. Durante este tiempo, el HTML se ha desarrollado de diferentes maneras. La Web depende de que los autores de páginas Web y las compañías compartan las mismas convenciones de HTML. Esto ha motivado el trabajo colectivo en las especificaciones del HTML.

El HTML 2.0 (noviembre de 1995) fue desarrollado bajo los auspicios de la Internet Engineering Task Force (IETF) para codificar lo que era la práctica común a finales de 1994. HTML+ (1993) y HTML 3.0 (1995) propusieron versiones mucho más ricas de HTML.

A pesar de no haber logrado nunca el consenso en las discusiones sobre estándares, estos borradores llevaron a la adopción de un número de nuevas características. Los esfuerzos del Grupo de Trabajo HTML del World Wide Web Consortium para codificar la práctica común en 1996 condujeron a HTML 3.2 (enero de 1997).

La mayoría de las personas están de acuerdo en que los documentos HTML deberían funcionar bien en diferentes navegadores y plataformas. Gracias a la interoperabilidad los proveedores de contenidos reducen gastos, ya que sólo deben desarrollar una versión de cada documento. Si este esfuerzo no se realiza, hay un riesgo mucho mayor de que la Web se convierta en un mundo propietario de formatos incompatibles, que al final acabaría por reducir el potencial comercial de la Web para todos los que forman parte de ella.

Cada versión de HTML ha intentado reflejar un consenso cada vez mayor entre los interlocutores de la industria, de modo que no se desperdicien las inversiones hechas por los proveedores de contenidos y que sus documentos no dejen de ser legibles a corto plazo.

El HTML ha sido desarrollado con la premisa de que cualquier tipo de dispositivo debería ser capaz de usar la información de la Web: PCs con pantallas gráficas con distintas resoluciones y colores, teléfonos móviles, dispositivos de mano, dispositivos de salida y entrada por voz, computadoras con anchos de banda grandes o pequeños, etc.

2.4.2.1 HTML 4

El HTML 4 desarrolla el lenguaje HTML con mecanismos para hojas de estilo, ejecución de scripts, marcos, objetos incluidos, soporte mejorado para texto de derecha a izquierda y direcciones mezcladas, tablas más ricas y mejoras en formularios, ofreciendo mejoras de accesibilidad para personas con discapacidades.

El HTML 4.01 es una revisión de HTML 4.0 que corrige errores e introduce algunos cambios desde la revisión anterior.

2.4.3 Internacionalización

Esta versión de HTML ha sido diseñada con la ayuda de expertos en el campo de la internacionalización, para que los documentos puedan ser escritos en cualquier idioma y fácilmente transportados por todo el mundo.

Un paso importante ha sido la adopción del estándar ISO/IEC:10646 como el conjunto de caracteres del documento para HTML. Este es el estándar más exhaustivo del mundo relacionado con la representación de caracteres internacionales, dirección del texto, puntuación, y otros aspectos de los idiomas del mundo.

HTML ofrece ahora un mayor soporte para diversos lenguajes humanos dentro de un documento. Esto permite un indexado más efectivo de los documentos por parte de los motores de búsqueda, tipografía de mayor calidad, mejor conversión de texto a voz, mejor separación de palabras, etc.

2.4.4 Accesibilidad

A medida que la comunidad de la Web crece y sus miembros diversifican sus habilidades, es crucial que las tecnologías sean apropiadas para sus fines específicos. El HTML se ha diseñado para hacer las páginas web más accesibles a aquéllos con limitaciones físicas. Los desarrollos de HTML 4 derivados de cuestiones de accesibilidad incluyen una mejor distinción entre la estructura y la presentación de un documento, aconsejando el uso de hojas de estilo en lugar de elementos y atributos de presentación de HTML para hacer las páginas aún más accesibles.

También dispone de mejores formularios, que incluyen la adición de teclas de acceso, la posibilidad de agrupar semánticamente los controles de un formulario, la posibilidad de agrupar las opciones SELECT semánticamente, y los rótulos activos. La posibilidad de codificar una descripción en texto de un objeto incluido (con el elemento OBJECT).

Asimismo se ha incluido un nuevo mecanismo de mapas de imágenes en el lado del cliente (el elemento MAP) que permite a los autores integrar vínculos de imagen y de texto. También se hace necesario incluir texto alternativo acompañando a las imágenes dentro del elemento IMG y a los mapas de imágenes dentro del elemento AREA.

Ahora los atributos title y lang pueden ser usados en todos los elementos, y se dispone de soporte para los elementos ABBR y ACRONYM. Al mismo tiempo se amplían los medios utilizables por hojas de estilo (tty, braille, etc.), y se mejoran las tablas, incluyendo títulos, grupos de columnas y mecanismos para facilitar su representación no visual.

2.4.5 Documentos compuestos

HTML ofrece ahora un mecanismo estándar para incluir objetos genéricos y aplicaciones dentro de documentos HTML. El elemento OBJECT (junto a los antiguos elementos IMG y APPLET, más específicos) proporciona un mecanismo para incluir imágenes, vídeo, sonido, fórmulas matemáticas, aplicaciones especializadas y otros objetos en un documento. También permite a los autores especificar una jerarquía de representaciones alternativas para los agentes de usuario que no soporten una representación específica.

2.4.6 Hojas de estilo

Las hojas de estilo simplifican el código HTML y liberan en gran medida al HTML de las responsabilidades de presentación. Esto da tanto a los autores como a los usuarios control sobre la presentación de los documentos: fuentes, alineación, colores, etc.

La información de estilo puede especificarse para elementos individuales o para grupos de elementos. La información de estilo puede especificarse en un documento HTML o en hojas de estilo externas.

El mecanismo para asociar una hoja de estilo con un documento es independiente del lenguaje de la hoja de estilo.

Antes de la llegada de las hojas de estilo, los autores tenían un control limitado sobre la representación. HTML 3.2 incluía un número de atributos y elementos que ofrecían control sobre la alineación, el tamaño de la fuente y el color del texto. Además los autores utilizaban las tablas y las imágenes como medio de organizar la presentación de sus páginas. El tiempo relativamente largo que necesitan los usuarios para actualizar sus navegadores hará que estas características sigan siendo usadas durante algún tiempo. Sin embargo, al ofrecer las hojas de estilo mecanismos de presentación más potentes, el World Wide Web Consortium declarará obsoletos en el futuro muchos de los elementos y atributos de presentación del HTML.

2.4.7 Ejecución de scripts

Gracias a los scripts, los autores pueden crear páginas web dinámicas (p.ej., "formularios inteligentes", que reaccionan a medida que los usuarios los rellenan) y utilizar el HTML para crear aplicaciones en red.

Los mecanismos proporcionados para incluir scripts en un documento HTML son independientes del lenguaje de programación de los scripts.

Referencia [19]

2.5 CSS

2.5.1 Descripción

CSS (Cascading Style Sheets, u Hojas de Estilo en Cascada) es la tecnología desarrollada por el World Wide Web Consortium (W3C) con el fin de separar la estructura de la presentación.

Fue creada en 1996 como estándar para las páginas web, el objetivo era solucionar un problema, y es que los navegadores existentes en aquel entonces (Netscape e Internet Explorer) estaban añadiendo etiquetas a la especificación de HTML, lo que dificultaba el entendimiento de una web en otros navegadores. Aunque ha tardado en ser soportada correctamente por muchos navegadores, hoy en día es utilizada por la mayoría de ellos.

2.5.2 Ventajas

Usando CSS evitamos hacer los archivos demasiado grandes y pesados (salvo el código requerido para las tablas anidadas y el añadido de características gráficas), y podemos definir el "estilo visual" de un sitio web entero sin necesidad de hacerlo etiqueta por etiqueta, para cada una de las páginas.

Por otro lado, trabajamos con estándares, y separamos hasta cierto punto la estructura (el código) de la presentación, logrando una manera más clara de trabajar.

Y lo que es más: en un sencillo documento CSS, definimos una plantilla gráfica para todo un sitio web, lo que permite que un cambio en el documento se produzca en todo el sitio web, de modo que con editar un documento CSS produces multitud de cambios que habría que realizar uno a uno en otro caso.

2.5.3 Historia

Las hojas de estilos aparecieron poco después que el lenguaje de etiquetas SGML, alrededor del año 1970. Desde la creación de SGML, se observó la necesidad de definir un mecanismo que permitiera aplicar de forma consistente diferentes estilos a los documentos electrónicos.

El gran impulso de los lenguajes de hojas de estilos se produjo con el boom de Internet y el crecimiento del lenguaje HTML para la creación de documentos. La guerra de navegadores y la

falta de un estándar para la definición de los estilos dificultaban la creación de documentos con la misma apariencia en diferentes navegadores.

El W3C propuso la creación de un lenguaje de hojas de estilos específico para el lenguaje HTML y se presentaron nueve propuestas. Las dos propuestas que se tuvieron en cuenta fueron la CHSS (Cascading HTML Style Sheets) y la SSP (Stream-based Style Sheet Proposal).

La propuesta CHSS fue realizada por Håkon Wium Lie y SSP fue propuesto por Bert Bos. Entre finales de 1994 y 1995 Lie y Bos se unieron para definir un nuevo lenguaje que tomaba lo mejor de cada propuesta y lo llamaron CSS (Cascading Style Sheets).

En 1995, el W3C decidió apostar por el desarrollo y estandarización de CSS y lo añadió a su grupo de trabajo de HTML. A finales de 1996, el W3C publicó la primera recomendación oficial, conocida como “CSS nivel 1”. A principios de 1997, el W3C decide separar los trabajos del grupo de HTML en tres secciones: el grupo de trabajo de HTML, el grupo de trabajo de DOM y el grupo de trabajo de CSS.

El 12 de Mayo de 1998, el grupo de trabajo de CSS publica su segunda recomendación oficial, conocida como “CSS nivel 2”. La siguiente recomendación, conocida como “CSS nivel 3”, continúa en desarrollo desde 1998 y hasta el momento sólo se han publicado borradores.

La adopción de CSS por parte de los navegadores ha requerido un largo periodo de tiempo. El mismo año que se publicó CSS 1, Microsoft lanzaba su navegador Internet Explorer 3.0, que disponía de un soporte bastante reducido de CSS. El primer navegador con soporte completo de CSS 1 fue la versión para Mac de Internet Explorer 5, que se publicó en el año 2000. Por el momento, ningún navegador tiene soporte completo de CSS 2.

De hecho, uno de los navegadores más utilizados, Internet Explorer 6, tiene un soporte limitado de CSS 2 y decenas de errores conocidos en la parte de CSS 2 que implementa, lo que dificulta la creación de páginas con un aspecto homogéneo entre diferentes navegadores. Los navegadores con mejor soporte de CSS 2 (incluso con soporte de algunas características de CSS 3) son Firefox (con su motor Gecko), Opera (con su motor Presto) y Safari/Konqueror (con su motor KHTML).

Desde la publicación de la versión CSS 2, se han añadido pequeñas correcciones de errores y algunas variaciones en el estándar, hasta llegar a la actual versión CSS 2.1.

Referencia [19]

Capítulo 3: Requisitos y Diseño

3.1 Introducción

El objetivo de este proyecto es diseñar e implementar una nueva base de datos para la ONG ASEM, puesto que la que usan actualmente tiene serios fallos de diseño y presenta muchos problemas técnicos a la hora de llevar a cabo actualizaciones o de realizar consultas.

Para ello, se procedió a analizar la documentación existente de la base de datos anterior, que consistía en la estructura de las tablas con los dominios de cada atributo. También se llevaron a cabo reuniones con representantes de la ONG, así como comunicación vía email para evaluar la evolución del proyecto y las posibles modificaciones que se podían llevar a cabo tanto en la información almacenada como en las funciones deseadas.

En este apartado del proyecto se hará un resumen del estado actual de la base de datos de la ONG, de los sistemas que poseen, así como de los objetivos que se persiguen con ese proyecto. Además de incluir las especificaciones dadas por la ONG para el desarrollo de la base de datos, la descripción del nuevo sistema a implementar, y los requisitos necesarios para poner en funcionamiento la nueva aplicación.

3.2 Estado actual

Actualmente la base de datos de que se dispone esta realizada en Access. Por las características de la misma, se puede deducir que se ha ido ampliando sin realizar ningún tipo de diseño previo, de modo que su estructura responde únicamente a las circunstancias que han ido surgiendo en cada momento.

Dichas ampliaciones han sido llevadas a cabo por personal sin conocimientos amplios de bases de datos, lo que ha provocado que la base de datos se haya vuelto compleja y muy ineficiente al tener tablas enormes con muchos campos vacíos y al mismo tiempo pequeñas tabas de un solo campo para valores booleanos.

En el diseño actual existen amplias redundancias en los datos, por ejemplo dispone de una tabla llamada “Datos Generales”, dicha tabla contiene todo tipo de datos, desde los datos personales a la pertenencia a grupos de usuarios, cada grupo de usuarios es un atributo en esta tabla, lo que implica que si se crea un grupo mas, es necesario añadir otro atributo a la tabla y dar un valor a todos los usuarios introducidos previamente.

A continuación podemos ver el diseño de la tabla “Datos Generales”, presente en la base de datos actual:

Tabla: Datos generales

Columnas

Nombre	Tipo	Tamaño
Id	Entero largo	4
Apellido 1º	Texto	50
Apellido 2º	Texto	50
Nombre	Texto	50
Fecha nacimiento	Fecha/Hora	8
Edad	Entero largo	4
Código Postal	Texto	6
Dirección	Texto	255
Localidad	Texto	255
Provincia	Texto	255
Teléfono	Texto	9
Móvil	Texto	9
e-mail	Texto	30
Profesión	Texto	255
Estudios	Texto	255
Escolarización	Texto	255
DNI/CIF	Texto	255
Carnet conducir	Texto	255
Coche	Texto	255
Sexo	Texto	255
Familia	Texto	255
Parentesco	Texto	255
Afectado	Sí/No	1
Certificado minusvalía	Sí/No	1
Grado minusvalía	Entero largo	4
Enfermedad	Texto	255
Fecha diagnóstico	Fecha/Hora	8
Control médico actual	Sí/No	1
Centro médico	Texto	255
Médico	Texto	255
Intervenciones quirúrgicas	Texto	255
Grado autonomía	Texto	255
Silla ruedas	Texto	255
Bono taxi	Sí/No	1
Rehabilitación	Sí/No	1
Centro rehablitación	Texto	255
Asociado ASEM	Sí/No	1
Grupo Padres	Sí/No	1
Grupo Adultos	Sí/No	1
Grupo Joven	Sí/No	1
Grupo Teatro	Sí/No	1
Grupo Vida Autónoma	Sí/No	1
Comisión Comunicación	Sí/No	1
Comisión Movilidad	Sí/No	1
Comisión Eventos	Sí/No	1
Comisión web	Sí/No	1
Colaborador Revista ASEM	Sí/No	1
Colaborador Ardilla	Sí/No	1
Voluntario	Sí/No	1
Colaborador	Sí/No	1
Junta Directiva	Sí/No	1
Cargo	Texto	255
Comité Expertos	Sí/No	1
Programa Apoyo Educativo	Sí/No	1
Programa Apoyo Hospitalario	Sí/No	1
Observaciones	Memo	-
Importe anual	Entero largo	4
Periodicidad	Entero largo	4
Importe cuota	Entero largo	4
Fechas pago	Texto	255
Cuenta bancaria	Texto	255
Al corriente pago	Sí/No	1

También podemos ver en el diseño actual, que en la misma tabla anterior, existen atributos “coche”, “bono taxi”, “carnet conducir”, “Estudios”, etc. que contienen cadenas de texto, y al

mismo tiempo existen tablas con los mismos nombres que esos atributos y que contienen asimismo cadenas de texto, a partir de lo cual podemos concluir que la información que contienen dichos atributos se encuentra duplicada, de forma que se pueda obtener a quien pertenece un coche y una lista de coches; pero al llevar a cabo esta estructura, no podemos garantizar que los datos de las dos tablas sean los mismos en un momento dado, con lo que cualquier consulta que realicemos devolverá información que no podemos confirmar, y al mismo tiempo si se realizar una comparación entre tablas, podemos encontrarnos con un conflicto de difícil solución, al ser imposible saber cuál es la información fiable.

Asimismo podemos observar que en la tabla “datos para envio correos” se vuelven a indicar datos personales, con lo que se duplica de nuevo la información sin forma alguna de controlar que dicha información sea coherente con la tabla “Datos Generales”. A todo esto podemos añadir que según avanzamos en el análisis del modelo actual vemos que también existe una tabla “Entrevistas”, en la cual se vuelven a introducir los datos personales de forma que podemos llegar a tener los datos personales de alguien en tres puntos distintos de la base de datos, lo cual resulta no solo muy poco práctico, si no que puede dar lugar a un elevado número de conflictos, puesto que resulta complicado saber cuál de todas las tablas contiene la información correcta.

A todos los problemas de conflictos que se pueden causar por la innecesaria información redundante, nos encontramos con un problema más serio aún, en lo que respecta al cumplimiento de la ley de protección de datos, dado que en caso de que un usuario desee modificar sus datos, resultará complicado asegurarse de que hemos modificado todos los datos y no solo una de las tablas con dicha información.

Debido a que la base de datos actual se encuentra desarrollada en Access, se presentan una serie de problemas de concurrencia en el acceso a los datos por parte de múltiples usuarios. Dicho problema se resuelve bloqueando el acceso al fichero que contiene la base de datos, pero de ese modo la aplicación pierde mucha funcionalidad, ya que solo puede ser usada por un usuario cada vez, lo cual no resulta una medida eficiente.

3.3 Objetivos

Dado el estado de la base de datos anterior, se ha optado por diseñar una base de datos completamente nueva desde el principio, tomando como referencia la anterior para la información que se deseaba guardar. También se han llevado a cabo varias reuniones con los miembros de la ONG, de forma que expusieran aquellas funciones que deseaban para la base de datos.

Puesto que se trata de una base de datos amplia, se optó por dividir el proyecto en secciones según su funcionalidad, separando por un lado las labores relacionadas con la administración de la ONG (cuotas, envío de emails, etc.) y por otro lado la parte de usuarios y enfermedades.

Se decidió desarrollar primero la parte de usuarios y dejar la parte administrativa para futuros desarrollos, de forma que la presente aplicación solo abarca la gestión de usuarios y enfermedades.

Tras revisar varios marcos de trabajo, se ha optado por desarrollar la aplicación haciendo uso del Framework CakePHP, dado que dispone de librerías que facilitarán el desarrollo, y además tiene fácil instalación y resulta muy portable.

De cara a facilitar el uso de la aplicación por parte de los usuarios, también se ha llevado a cabo un diseño completo para el interfaz de la aplicación, dicho interfaz web ha sido creado siguiendo los estándares de accesibilidad lo más posible, aunque por motivos de rendimiento, es necesario que el navegador habilite javascript, de modo que la aplicación pueda cargar la información nueva prescindiendo de toda la que ya ha cargado previamente.

Al llevar a cabo el diseño del interfaz, se ha pretendido que la aplicación sea muy sencilla, de forma que no se requiera demasiada formación para los encargados de manejar los datos. Dado que se trata de personas sin formación específica para el manejo de bases de datos, se ha pretendido que el diseño de la aplicación sea intuitivo y no cause confusión, de manera que puedan familiarizarse rápidamente con el funcionamiento de la aplicación y sacar el máximo partido posible a la misma.

3.4 Requisitos de la ONG

Al tratarse de una aplicación a petición de una ONG, se deben seguir una serie de requisitos impuestos por la misma, dichos requisitos se proporcionaron de varias formas. Mediante reuniones con representantes de la misma, mediante correos electrónicos, así como por el estudio de la estructura de la base de datos ya existente.

Durante las reuniones se proporcionó documentación relativa a la base de datos anterior, como ya se indica en el estado actual, además se hablo del manejo de información que tenía la ONG en aquel momento y del que esperaban tener una vez se diera por finalizado el proyecto. Entre los usos que se pretendían conseguir, estaba la obtención de datos estadísticos sobre el número de personas con una determinada enfermedad o pertenecientes a un determinado grupo.

Es necesario tener en cuenta en todo momento, que puesto que almacenaremos datos personales, es necesario cumplir la ley orgánica de protección de datos (LOPD).

3.4.1 Especificaciones Iniciales

Primero especificaremos aquellas decisiones de diseño que han sido proporcionadas por la ONG al inicio del proyecto y que son la base del proyecto.

- *La base de datos debe contener al menos la misma información que la ya existente*.

Puesto que ya se dispone de una base de datos previa, la nueva base de datos debe almacenar la misma información, aunque varíe la estructura y la forma de almacenar dicha información.

Se ha diseñado desde cero la nueva base de datos, optimizando las tablas de forma que la información se encuentre lo más ordenada posible. Tras dichos cambios se ha optimizado el acceso a los datos y el espacio ocupado por la base de datos, al tener la cantidad mínima posible de campos vacíos en las tablas. También se ha reducido al mínimo la redundancia de datos de forma que el sistema sea más consistente y de esta forma la aplicación no tenga problemas de coherencia en un futuro.

- *Debe ser posible hacer consultas de los datos de enfermos, sin acceder a los datos personales de los mismos*.

Debido a la ley orgánica de protección de datos (LOPD), es necesario separar los datos de los enfermos de sus datos personales. Por este motivo se han creado dos tablas independientes en la base de datos, pfc-persona y pfc-enfermo. La primera almacena los datos personales del usuario sea o no un enfermo. La segunda solo almacena los datos médicos de los enfermos, permitiendo acceder a los mismos sin ver los datos personales.

De este modo es posible realizar consultas relativas a los enfermos sin acceder a información personal, también de esta forma resulta posible solucionar el problema de redundancia de datos presente en la base de datos anterior, sea cual sea el usuario de la aplicación, sus datos se encontrarán en la tabla persona, independientemente de que el usuario sea un enfermo o no lo sea.

- *Acceso a la base de datos desde Internet.*

En las especificaciones iniciales de la ONG, se indicaba la necesidad de que la aplicación fuera accesible desde cualquier lugar. Aunque posteriormente éste requisito fue retirado, la decisión de implementar la aplicación en una plataforma web, facilita que en un futuro se puedan llevar a cabo con sencillez los cambios necesarios en la aplicación para hacer los datos accesibles desde internet. Uno de los cambios necesarios para esto sería la implementación de las listas de control de acceso en la aplicación, de forma que se pueda confirmar la identidad de aquellos usuarios que accedan a los datos en la aplicación.

Es importante recordar que en el momento de implementar el acceso a internet, la seguridad de la aplicación seguirá recayendo sobre la contraseña de los usuarios que tengan acceso, por lo cual es conveniente que hagan uso de las recomendaciones de seguridad.

3.4.2 Especificaciones posteriores

Muchas de las especificaciones de la aplicación se han recibido durante el proceso de desarrollo de la misma, mediante reuniones con los representantes de la ONG o en correos electrónicos. En este apartado detallaremos aquellos requisitos que fueron especificados por la ONG durante el desarrollo del proyecto.

- *Acceso en red local.*

Pese a que en un principio se requirió que la aplicación fuese accesible desde cualquier lugar de la red. En una reunión posterior, se indicó que la aplicación solo debía ser accesible desde la red local.

El resultado de dicha especificación fue que se pospuso para una futura ampliación del proyecto la elaboración de un método de control de acceso, al no ser accesible desde el exterior de la red local. Esta especificación requiere que el servidor en el que se monte la base de datos se configure para permitir únicamente accesos desde su red y no de redes externas.

- *Acceso desde múltiples puntos a la aplicación al mismo tiempo.*

La base de datos anterior, al haber sido realizada en Microsoft Access, tenía el problema de ser accesible únicamente por un solo usuario cada vez, siendo muy ineficiente para consultas e inserción de datos. Por este motivo, una de las especificaciones dadas por la ONG durante la elaboración del proyecto, es la posibilidad de que varias personas puedan realizar consultas, inserciones y modificaciones en la base de datos al mismo tiempo.

Dicho requisito queda cubierto por el entorno en que se realiza la aplicación, mysql incluye por defecto el control de concurrencia, además de soportar un número elevado de transacciones simultaneas, y dado que solo se espera un pequeño número de ordenadores conectados a la aplicación, no hay riesgo de sobrecarga en la base de datos, o de superar la capacidad de la red.

- *Fácil creación de grupos de usuarios*

En la base de datos actual, los grupos de usuarios se han incluido como atributos de la tabla “Datos Generales”, atributos con un valor booleano que indica si el usuario pertenece o no a dicho grupo. Dicha implementación es muy ineficiente cuando se maneja un volumen de datos medianamente alto, puesto que nos vemos en la necesidad de recorrer todas las personas verificando si el valor otorgado al atributo correspondiente es verdadero o falso. A lo que hay que añadir las limitaciones que impone, al ser necesario modificar la estructura de la base de datos para añadir algo tan simple como un grupo de usuarios.

Por este motivo se ha creado una tabla grupos que se relaciona con la tabla de personas, de este modo una persona puede estar incluida en varios grupos sin que almacenemos en el

sistema montones de valores sin utilidad, para saber que usuarios pertenecen a un grupo, solo hace falta consultar la tabla de relaciones en busca del identificador de dicho grupo

- *Control de nombres extranjeros*

En un principio por hacer más robusta la base de datos, se puso como obligatorio la introducción de al menos el nombre y los 2 apellidos. Durante una reunión con representantes de la ONG, indicaron que aquellas personas de origen extranjero, podían no tener un segundo apellido, como es el caso de muchos anglosajones, y por dicho motivo se retiró la obligatoriedad del segundo apellido.

- *Identificación mediante el DNI.*

En un principio se controlaba la introducción del DNI en la aplicación, lo que hacía imposible que un usuario extranjero introdujera su pasaporte por tener un formato distinto, a este hecho se añadía que muchos niños no disponen de DNI. Este problema se solventó al permitir la introducción de cadenas alfanuméricas distintas del DNI y también permitiendo los valores nulos. De esta forma se podían introducir a niños pese a no tener DNI o pasaporte.

Este problema ha sido uno de los más molestos en la aplicación, ya que aunque la tabla personas posee un identificador único de cada usuario, esperaba poder establecer el DNI como clave alternativa, pero tras documentarme un poco pude averiguar que, aunque es infrecuente, es posible encontrar personas con el mismo DNI, lo cual me impedía usar esta clave como alternativa.

- *Bloqueo de las tablas durante las modificaciones.*

Durante una reunión con la ONG, se comentó la posibilidad de bloquear las tablas de forma que mientras se realizaba una modificación, no fuera posible realizar ninguna otra sobre dicha información.

Como ya hemos indicado previamente, la aplicación solo se ejecutará en red local, a ese punto debemos añadir que los usuarios que se espera que trabajen sobre dicha aplicación son tres, y se encuentran en la misma sala. Con todo lo anterior, los beneficios que se pueden obtener al bloquear la tabla son muy inferiores a los perjuicios, dado que nos encontraríamos en una situación semejante a la de partida, en la que solo un usuario podría trabajar cada vez. Por este motivo la funcionalidad fue descartada.

3.5 Descripción del sistema

El sistema consiste en una aplicación Web desarrollada en PHP, HTML, CSS y javascript mediante el Framework CakePHP, que a su vez accede a una base de datos desarrollada en mysql.

Gracias a CakePHP, el manejo de javascript en la aplicación es muy sencillo y prácticamente se limita a indicar al framework que queremos hacer uso de AJAX. Gracias al uso de AJAX (Asynchronous JavaScript And Xml) podemos hacer que las cargas de la información sean más rápidas, al actualizar solo aquella información que el usuario ha pedido y no toda la información de la página web.

En el caso de nuestra aplicación, el uso de AJAX permitirá que el menú y la cabecera se mantengan durante la navegación, limitando los cambios a la parte de contenido en la cual se muestra la información y donde el usuario debe introducir las consultas, inserciones o modificaciones de los datos.

La aplicación permite que el usuario pueda insertar nuevos registros, modificar los ya existentes, o simplemente consultar registros, de forma simple y sin necesidad de conocimientos sobre la estructura de la base de datos, la propia aplicación se encarga de verificar que los datos introducidos por el usuario son correctos, o al menos cumplen con el formato apropiado, y en caso de no ser así, indica al usuario cual es el error cometido, para que de esa forma pueda solucionarlo antes de provocar una inconsistencia en la base de datos. En caso de que los datos que se pretendan introducir provoquen una inconsistencia y no se modifiquen, el sistema no permitirá al usuario realizar dicha operación para mantener la integridad de la base de datos.

El diseño realizado para la aplicación, permite agrupar a las personas de forma simple, por familias o por grupos creados por el usuario, de esta forma se cubre uno de los mayores problemas del diseño anterior, que requería modificar la estructura de la base de datos cada vez que se pretendía crear un grupo de usuarios nuevo. Al mismo tiempo gracias al framework utilizado, en un futuro podría incluirse en la aplicación el control de usuarios mediante ACLs (Listas de Control de Acceso) sin requerir modificaciones serias de la estructura de la base de datos y realizando cambios mínimos en la aplicación.

3.6 Recursos del sistema

Conforme a la descripción del sistema que hemos visto en el apartado anterior, los recursos necesarios para el funcionamiento de la aplicación no son muchos, ya que la base de datos no va a tener un tamaño que requiera un soporte especial, el framework utilizado tiene un tamaño reducido y el acceso a la aplicación se puede hacer desde cualquier ordenador que soporte navegación Web.

Se requiere una red local en la que se encuentren los siguientes elementos, el sistema operativo de los ordenadores presentes en la red es indiferente, ya que la aplicación se ha implementado para ser usada por cualquier plataforma.

Usuario	**Software**
(PC)Servidor Web (Al ser una aplicación pequeña basta con uno)	Apache PHP MySQL CakePHP
(PC)Cliente (Tantos como se desee)	Navegador Web JavaScript (opcional)

Capítulo 4: Implementación

4.1 Introducción

En este apartado se desarrolla la implementación del proyecto. El diseño de las tablas se ha efectuado en base a la documentación aportada por la ONG y a reuniones mantenidas con sus representantes. En base a dichas reuniones y a esa documentación se preparó un diseño relacional, que posteriormente fue modificado para incluir más información requerida por la ONG.

Dado que el sistema no se prevé que vaya a tener un gran tráfico, se ha normalizado la base de datos lo más posible, de este modo se obtiene una estructura más clara y comprensible, con las mínimas redundancias. En el caso de que en un futuro la aplicación fuera a recibir mucho tráfico, posiblemente sería necesario desnormalizar las tablas, para de ese modo obtener un rendimiento mayor en las transacciones.

La implementación se divide en dos partes, por un lado el diseño de las tablas de la base de datos y por otro lado la codificación de la aplicación.

4.2 Tablas

En esta sección se presentará una relación de las tablas que se han diseñado para este proyecto. En cada tabla se indica cual es el campo identificador y cuáles de los atributos son obligatorios y cuales opcionales.

Asimismo se da una descripción de la función de cada tabla y los tipos de datos de cada campo.

Notación utilizada:

Atributo	**Presentación**
Clave Primaria	**Negrita**
Clave Ajena	*Cursiva*
Clave Alternativa	<u>Subrayada</u>
Atributos Opcionales	Asterisco*

4.2.1 PFC-Persona

Contiene los datos personales, toda persona presente en la aplicación, sea o no un enfermo, tendrá una entrada en esta tabla. Salvo nombre, apellido1 y email, los demás campos son opcionales. El campo id_p es un identificador de tabla autoincremental.

Nombre del Atributo	Descripción	Tipo de Dato
id_p	Identificador de la tabla	INTEGER(8)
dni*	DNI	VARCHAR(9)
nombre	Nombre	VARCHAR(50)
apellido1	Primer apellido	VARCHAR(50)
apellido2*	Segundo apellido	VARCHAR(50)
fecha_nacimiento*	Fecha de nacimiento	DATE
movil*	Teléfono móvil	VARCHAR(20)
email	Correo electrónico	VARCHAR(50)
via*	Dirección	VARCHAR(50)
codigo_postal*	Código postal	INTEGER(5)
localidad*	Localidad	VARCHAR(50)
provincia*	Provincia	VARCHAR(50)
pais*	País	VARCHAR(50)
telefono*	Teléfono fijo	VARCHAR(20)
estudios*	Nivel de estudios	VARCHAR(50)
profesion*		VARCHAR(50)

4.2.2 PFC-Familia

Contiene los nombres de las familias, de forma que se puedan agrupar los usuarios en función de la familia a la que pertenezcan. Es obligatorio el nombre y el campo id_f es un identificador de tabla autoincremental.

Nombre del Atributo	Descripción	Tipo de Dato
id_f	Identificador de la tabla	INTEGER(8)
nombre	Nombre identificativo para la familia	VARCHAR(50)

4.2.3 PFC-Grupo

Contiene los datos de los grupos de usuarios, únicamente nombre y descripción. Ambos campos son obligatorios. El campo id_g es un identificador de tabla autoincremental.

Nombre del Atributo	Descripción	Tipo de Dato
id_g	Identificador de la tabla	INTEGER(8)
nombre	Nombre identificativo para el grupo de usuarios	VARCHAR(50)
descripcion	Descripción de la función del grupo	VARCHAR(100)

4.2.4 PFC-Enfermo

Contiene los datos médicos de cada enfermo, se encuentran separados de los datos personales para cumplir la ley de protección de datos, así como para hacer más eficiente la base de datos. Salvo el certificado de enfermedad, todos los campos son obligatorios. El campo id_p referencia a los datos personales del enfermo, por si resultan necesarios. El campo id_en es un identificador de tabla autoincremental.

Nombre del Atributo	Descripción	Tipo de Dato
id_en	Identificador de la tabla	INTEGER(8)
id_p	Clave ajena que referencia a la persona.	INTEGER(8)
certificado*	Certificado de la enfermedad en caso de que exista	VARCHAR(9)
grado_minusvalia	Indicador del grado de minusvalía	VARCHAR(50)
grado_autonomia	Indicador del grado de autonomía	VARCHAR(50)
bono_taxi	Indicador de si se tiene o no un bono taxi	VARCHAR(2)
silla_ruedas	Tipo de silla de ruedas	VARCHAR(50)

4.2.5 PFC-Enfermedades

Contiene los datos de cada enfermedad. Cada enfermo se asocia a una o más enfermedades, que a su vez se asocian a sus síntomas y tratamientos. Salvo el campo familiar todos son obligatorios. El campo id_e es un identificador de tabla autoincremental.

Nombre del Atributo	Descripción	Tipo de Dato
id_e	Identificador de la tabla	INTEGER(8)
nombre	Nombre común de la enfermedad	VARCHAR(50)
nombre_c	Nombre científico de la enfermedad	VARCHAR(200)
aparicion	Edad de aparición de la enfermedad	INTEGER(2)
hereditaria	Indica si es o no hereditaria.	VARCHAR(2)
antecedentes		VARCHAR(2)
familiar*		VARCHAR(50)
diagnostico	Tipo de diagnóstico	'electro', 'biopsia', 'analisis'

4.2.6 PFC-Sintomas

Contiene los datos de cada síntoma, se relacionan con las distintas enfermedades, de forma que no se repitan por cada enfermedad. Son obligatorios el nombre y la descripción. El id_s es un identificador de tabla autoincremental.

Nombre del Atributo	Descripción	Tipo de Dato
id_s	Identificador de la tabla	INTEGER(8)
nombre	Nombre común del síntoma	VARCHAR(50)
descripción	Descripción del síntoma	VARCHAR(100)

4.2.7 PFC-Tratamientos

Contiene los datos de cada tratamiento, se relacionan con las distintas enfermedades, de forma que no se repitan por cada enfermedad. Son obligatorios el nombre y la descripción. El id_t es un identificador de tabla autoincremental.

Nombre del Atributo	Descripción	Tipo de Dato
id_t	Identificador de la tabla	INTEGER(8)
nombre	Nombre común del tratamiento	VARCHAR(50)
descripción	Descripción del tratamiento	VARCHAR(100)

4.2.8 PFC-Enfermo-Enfermedad

Contiene los datos que relacionan a cada enfermo con su enfermedad. Los campos id_en y id_e referencian al enfermo y a la enfermedad respectivamente.

Nombre del Atributo	Descripción	Tipo de Dato
id_en	Identificador de la tabla/clave ajena	INTEGER(8)
id_e	Identificador de la tabla/clave ajena	INTEGER(8)
fecha_diagnostico*	Fecha en la que le fue realizado el diagnóstico	DATE
control*	Descripción del tratamiento	VARCHAR(50)
centro	Nombre del centro donde recibe tratamiento	VARCHAR(50)
intervenciones	Número de intervenciones practicadas	INTEGER(2)

4.2.9 PFC-Familia-Persona

Contiene los datos que relacionan a cada persona con su familia. Los campos id_f y id_p referencian a la familia y a la persona respectivamente.

Nombre del Atributo	Descripción	Tipo de Dato
id_f	Identificador de la tabla/clave ajena	INTEGER(8)
id_p	Identificador de la tabla/clave ajena	INTEGER(8)

4.2.10 PFC-Grupo-Persona

Contiene los datos que relacionan a cada persona con un grupo. Los campos id_g y id_p referencian al grupo y a la persona respectivamente.

Nombre del Atributo	Descripción	Tipo de Dato
id_g	Identificador de la tabla/clave ajena	INTEGER(8)
id_p	Identificador de la tabla/clave ajena	INTEGER(8)

4.2.11 PFC-Sintoma-Enfermedad

Contiene los datos que relacionan a cada síntoma con una enfermedad. Los campos id_s y id_e referencian al síntoma y a la enfermedad respectivamente.

Nombre del Atributo	Descripción	Tipo de Dato
id_s	Identificador de la tabla/clave ajena	INTEGER(8)
id_e	Identificador de la tabla/clave ajena	INTEGER(8)

4.2.12 PFC-Tratamiento-Enfermedad

Contiene los datos que relacionan a cada tratamiento con una enfermedad. Los campos id_t y id_e referencian al tratamiento y a la enfermedad respectivamente.

Nombre del Atributo	Descripción	Tipo de Dato
id_t	Identificador de la tabla/clave ajena	INTEGER(8)
id_e	Identificador de la tabla/clave ajena	INTEGER(8)

4.3 Consultas

Al hacer uso del Framework CakePHP, los accesos a la base de datos quedan encapsulados y no es necesario realizar la programación de las consultas.

4.4 Diagrama del modelo de la base de datos

En la siguiente imagen se muestran las relaciones entre las distintas tablas. Lo que representa este esquema es que una persona puede pertenecer a mas de una familia y al mismo tiempo en una familia pueden pertenecer varias personas, lo que indica una relación N:M y explica la existencia de la tabla intermedia. En el caso de los grupos sucede igual. El motivo por el que se han separado familias y grupos es más que nada por simplicidad en el diseño, podrían incluirse las familias como un tipo de grupos.

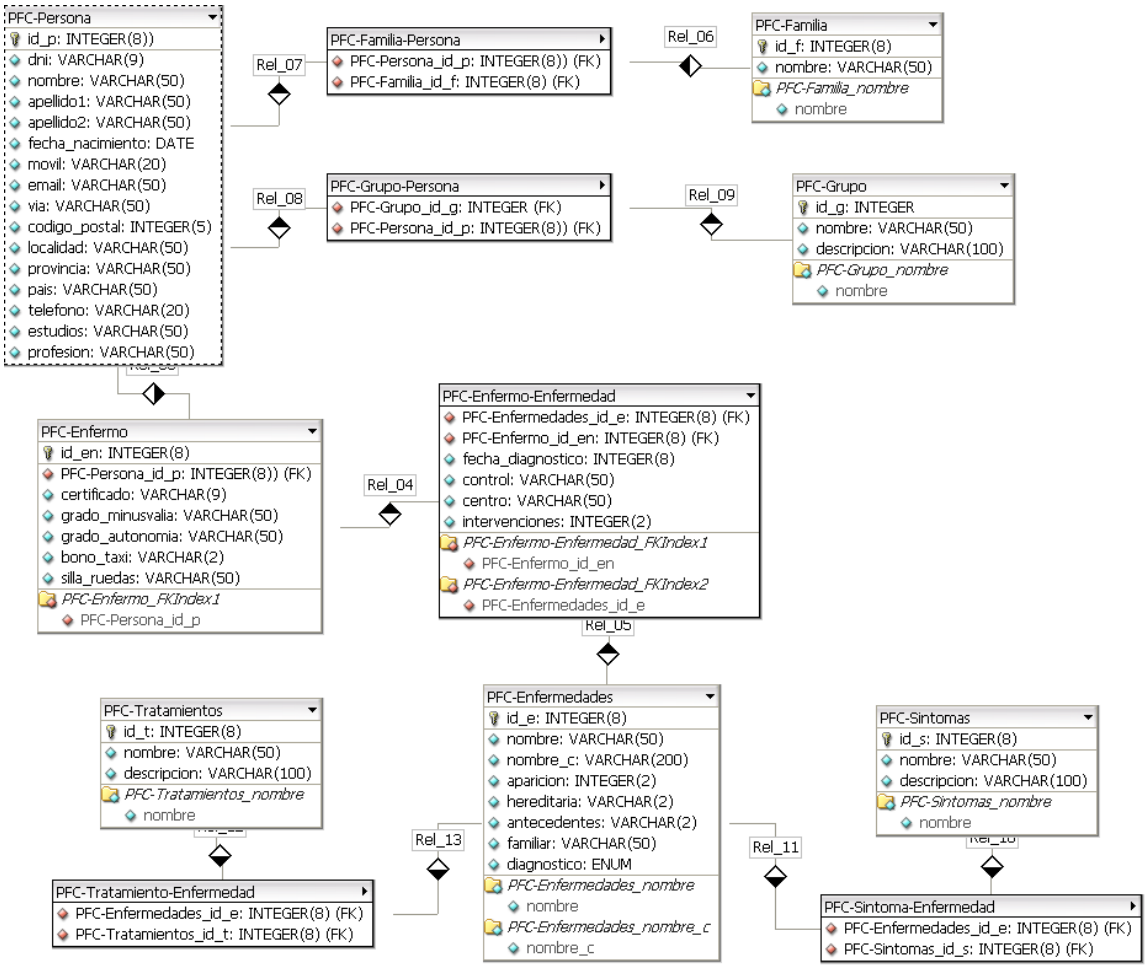
La relación entre persona y enfermo es de 0:1, puesto que una persona puede estar dada de alta en la aplicación sin ser enfermo. La siguiente relación entre enfermo y enfermedad es asimismo una N:M al poder existir un enfermo con varias enfermedades y poder tener una misma enfermedad muchos enfermos.

Los tratamientos y los síntomas se relacionan únicamente con las enfermedades pero no con los enfermos. Esto hace que se pueda averiguar que síntomas y tratamientos tiene una enfermedad, pero no es posible saber cuáles de ellos tiene exactamente un enfermo. Ambas son relaciones N:M al poder tener varias enfermedades los mismos síntomas y tratamientos, y al mismo tiempo dichos síntomas y tratamientos pueden estar relacionados con muchas enfermedades distintas.

La relación entre tratamientos, síntomas y enfermos queda para un futuro desarrollo de la aplicación. Siendo ambas relaciones N:M que requerirían tablas intermedias.

Todas las actualizaciones y los borrados son en cascada, eso implica que al borrar una persona, se borra el enfermo asociado a ella, y todas las relaciones familiares y de grupo que tenga. Obviamente no se borran ni la Familia ni los grupos a los que pertenezca, únicamente la relación entre Persona y Familia así como entre Persona y Grupo.

De este modo obtenemos una mayor integridad en la base de datos. Al mantener la coherencia de la misma.


PFC-Persona
id_p: INTEGER(8))
dni: VARCHAR(9)
nombre: VARCHAR(50)
apellido1: VARCHAR(50)
apellido2: VARCHAR(50)
fecha_nacimiento: DATE
movil: VARCHAR(20)
email: VARCHAR(50)
via: VARCHAR(50)
codigo_postal: INTEGER(5)
localidad: VARCHAR(50)
provincia: VARCHAR(50)
pais: VARCHAR(50)
telefono: VARCHAR(20)
estudios: VARCHAR(50)
profesion: VARCHAR(50)
Rel_07
PFC-Familia-Persona
PFC-Persona_id_p: INTEGER(8)) (FK)
PFC-Familia_id_f: INTEGER(8) (FK)
Rel_06
PFC-Familia
id_f: INTEGER(8)
nombre: VARCHAR(50)
PFC-Familia_nombre
nombre
Rel_08
PFC-Grupo-Persona
PFC-Grupo_id_g: INTEGER (FK)
PFC-Persona_id_p: INTEGER(8)) (FK)
Rel_09
PFC-Grupo
id_g: INTEGER
nombre: VARCHAR(50)
descripcion: VARCHAR(100)
PFC-Grupo_nombre
nombre
PFC-Enfermo
id_en: INTEGER(8)
PFC-Persona_id_p: INTEGER(8)) (FK)
certificado: VARCHAR(9)
grado_minusvalia: VARCHAR(50)
grado_autonomia: VARCHAR(50)
bono_taxi: VARCHAR(2)
silla_ruedas: VARCHAR(50)
PFC-Enfermo_FKIndex1
PFC-Persona_id_p
Rel_04
PFC-Enfermo-Enfermedad
PFC-Enfermedades_id_e: INTEGER(8) (FK)
PFC-Enfermo_id_en: INTEGER(8) (FK)
fecha_diagnostico: INTEGER(8)
control: VARCHAR(50)
centro: VARCHAR(50)
intervenciones: INTEGER(2)
PFC-Enfermo-Enfermedad_FKIndex1
PFC-Enfermo_id_en
PFC-Enfermo-Enfermedad_FKIndex2
PFC-Enfermedades_id_e
Rel_05
PFC-Tratamientos
id_t: INTEGER(8)
nombre: VARCHAR(50)
descripcion: VARCHAR(100)
PFC-Tratamientos_nombre
nombre
PFC-Tratamiento-Enfermedad
PFC-Enfermedades_id_e: INTEGER(8) (FK)
PFC-Tratamientos_id_t: INTEGER(8) (FK)
Rel_13
PFC-Enfermedades
id_e: INTEGER(8)
nombre: VARCHAR(50)
nombre_c: VARCHAR(200)
aparicion: INTEGER(2)
hereditaria: VARCHAR(2)
antecedentes: VARCHAR(2)
familiar: VARCHAR(50)
diagnostico: ENUM
PFC-Enfermedades_nombre
nombre
PFC-Enfermedades_nombre_c
nombre_c
Rel_11
PFC-Sintomas
id_s: INTEGER(8)
nombre: VARCHAR(50)
descripcion: VARCHAR(100)
PFC-Sintomas_nombre
nombre
PFC-Sintoma-Enfermedad
PFC-Enfermedades_id_e: INTEGER(8) (FK)
PFC-Sintomas_id_s: INTEGER(8) (FK)

4.5 Implementación del Código

En este apartado se muestran las peculiaridades del código php, HTML y CSS de la aplicación.

4.5.1 CSS

El estilo visual de la aplicación recae por completo en las hojas de estilo en cascada (CSS). Mediante identificadores en los contenedores (div).

Gran parte de la complejidad de la CSS está en el menú:

Esta parte sitúa el menú en la página.

```
/* menu */
#Menu{
        position:relative;
        float: left;
        text-align: left;
        margin-top: 7px;
        height: 100%;
        width: 200px;
}
```

En este trozo se da color a los hipervínculos presentes en el menú.

```
#Menu a {
        color:#666666;
}
```

El siguiente código define el estilo de la lista y quita el subrayado de los hipervínculos para hacer más homogéneo el menú. También pone color a los elementos del menú que se encuentran en la lista.

```
#Menu ul {
        list-style-type: square;
        text-decoration: none;
}

#Menu ul li{
        background-color:#FFFFFF;
}

#Menu ul li a{
        background-color:#FFFFFF;
}
```

A continuación se indican las propiedades de las listas internas del menú. Las que aparecen en el interior de cada sección.

```
#Menu ul ul{
        padding-left: 0px;
        margin-top: 3px;
        background-color:;
        border-left:1px solid #ccc;
}

#Menu ul ul li{
        line-height:20px;
        border-bottom:1px solid #ccc;
        list-style:none;
        padding-left:5px;
}

#Menu ul ul ul{
        margin-bottom: -1px;
        border-left:0px solid #ccc;
}
```

Otras partes de la hoja de estilos, tienen como función situar cada sección de la página en el lugar correcto de la misma. Así como definir un estilo común para toda la web.

Para empezar tenemos el cuerpo de la página, que define el tipo de texto a usar, así como los colores y fondos.

```
body {
        margin:0;
        font-family:verdana,arial,helvetica,sans-serif;
        font-size:62.5%;
        background-image: url(./fondo_body.jpg);
        background-position: center;
        background-repeat: repeat-y;
        background-color:#E9E9E9;
        text-align: center;
        color:#666666;
}
```

También tenemos unos atributos definidos para las imágenes que se muestren en la página:

```
img{
        border:0;
        margin: auto;
}

.centrar-imagen {
        text-align: center;
}
```

Hay establecido un marco dentro del cuerpo de la página, que permite centrar el contenido sea cual sea la resolución de pantalla.

```
#Marco {
	position:relative;
	font-size:1.1em;
	width:780px;
	margin:0 auto;
	height:auto;
	margin-bottom:45px;
}
```

Dentro del marco se establece una cabecera para la página. En dicha cabecera se ha introducido el logotipo, así como un enlace al index en todo momento.

```
/* cabecera */
#Cabecera{
	height: 120px;
	position:relative;
	margin: auto;
}
```

Para separar el menú y la cabecera del contenido de la página tenemos el div contenido. En el cuál se van cargando todas las acciones que se realizan.

```
/* contenido */
#Contenido{
	position:relative;
	float: left;
	height: 100%;
	padding-left: 19px;
	width: 550px;
}
```

4.5.2 PHP

Al hacer uso del Framework CakePHP, el código de la aplicación se encuentra dividido en varias partes:

4.5.2.1 Modelo

Por un lado se encuentran los modelos de datos, en ellos se indica la tabla que contiene los datos del modelo, las reglas de validación del mismo, y todas las relaciones que existan entre ese modelo y otros.

Un ejemplo de cuál sería la estructura de un modelo sería la siguiente:

```
class Enfermedad extends AppModel {
	var $name = 'Enfermedad';
	var $useTable = "pfc-enfermedades";
	var $primaryKey = 'id_e';
	var $validate = array(
		'nombre' => array(
			'ruleName' => array(
				'rule' => 'notEmpty',
				'message' => 'Por favor ingrese un nombre válido'
			),
			'ruleName2' => array(
				'rule' => 'isUnique',
				'message' => 'Ese nombre ya existe'
			)
		),
		'nombre_c' => array(
			'ruleName' => array(
				'rule' => 'notEmpty',
				'message' => 'Por favor ingrese un nombre válido'
			),
			'ruleName2' => array(
				'rule' => 'isUnique',
				'message' => 'Ese nombre ya existe'
			)
		),
		'aparicion' => array(
			'rule' => 'notEmpty',
			'message' => 'Por favor ingrese una edad de aparición válida, si
no conoce ninguna ponga 0'
		),
		'hereditaria' => array(
			'rule' => 'notEmpty',
			'message' => 'Por favor seleccione un valor válido, si no lo sabe
ponga NO'
```

```
            )
        );


        var $hasAndBelongsToMany = array(
         'Enfermo' =>
            array('className'            => 'Enfermo',
                 'joinTable'             => 'pfc-enfermo-enfermedad',
                 'foreignKey'            => 'id_e',
                 'associationForeignKey'  => 'id_en'
            ),
         'Sintoma' =>
            array('className'            => 'Sintoma',
                 'joinTable'             => 'pfc-sintoma_enfermedad',
                 'foreignKey'            => 'id_e',
                 'associationForeignKey'  => 'id_s'
            ),
         'Tratamiento' =>
            array('className'            => 'Tratamiento',
                 'joinTable'             => 'pfc-tratamiento_enfermedad',
                 'foreignKey'            => 'id_e',
                 'associationForeignKey'  => 'id_t'
            )
         );

}
```

En el código anterior podemos ver como la clase Enfermedad extiende de AppModel, que le proporciona los atributos necesarios para el modelo.

Se puede observar como el atributo name define el nombre del modelo, asimismo, el atributo “useTable” indica cuál será la tabla a usar, en caso de no indicarse se buscaría la tabla con un nombre equivalente al plural del modelo.

Del mismo modo, la variable “primaryKey” indica la clave primaria de la tabla. En caso de no especificarse, el modelo buscaría como clave primaria el campo “id”.

Podemos ver como se implementan las reglas de validación de datos, que permiten verificar que los campos no estén vacíos y que no se dupliquen entradas. De este modo evitamos tener que manejar las excepciones de la base de datos, al indicar al usuario el error antes de producirse el envío del formulario. Existe un gran número de reglas de validación ya predefinidas, pero en caso de ser necesario es posible escribir nuevas reglas personalizadas para la aplicación.

Otra característica del modelo de datos, es que en él puedes indicar las relaciones entre distintas modelos y en función de la relación que exista entre los modelos la aplicación reaccionará de distinta forma.
En el modelo anterior podemos ver, que el modelo enfermedad mantiene relaciones N:M con los modelos de Enfermo, Síntoma y Tratamiento. Existen varias relaciones posibles entre modelos.

hasOne: Se trata de una relación 0:1, como la que existe entre una persona y un enfermo. Se usa en el modelo referenciado, es decir, la clave ajena se encuentra en el Enfermo. Necesita saber qué modelo le referencia, el nombre de la clase, y la clave ajena que está usando. El siguiente código se encuentra en el modelo Persona:

```
var $hasOne = array('Enfermo'=> array(
                        'className' => 'Enfermo',
                        'foreignKey' => 'id_p'
                ));
```

belongsTo: Se trata del otro extremo de una relación 0:1, como la que existe entre una persona y un enfermo. Necesita que se indique el modelo al que hace referencia, el nombre de la clase y la clave ajena que usa. El siguiente código se encuentra presente en el modelo Enfermo.

```
var $belongsTo = array('Persona'=> array(
                        'className' => 'Persona',
                        'foreignKey' => 'id_p'
                ));
```

hasMany: Se trata del extremo referenciado de una relación 0:N, en el otro extremo se situaría una variable belongsTo. No se ha hecho uso de este tipo de relación en la aplicación.

hasAndBelongsToMany: Se trata de una relación N:M, es la que más usada en la aplicación. Por las propias características de las relaciones N:M, esta variable se encuentra en ambos modelos, indicando el nombre del otro modelo de la relación, la tabla en la que se cruzan las referencias y las claves ajenas que las relacionan. El siguiente código se encuentra presente en el modelo Enfermo y lo relaciona con el modelo Enfermedad.

```
var $hasAndBelongsToMany = array(
        'Enfermedad' =>
            array('className'            => 'Enfermedad',
                  'joinTable'             => 'pfc-enfermo-enfermedad',
                  'foreignKey'            => 'id_en',
                  'associationForeignKey'  => 'id_e'
            )
        );
```

4.5.2.2 Controlador

El controlador es la parte de la aplicación en la que se implementa la lógica de la misma. En CakePHP, el nombre del controlador tiene un formato específico, que viene a ser el nombre del modelo, en plural y añadiendo al final “_controller”. Eso hace que nuestro controlador para enfermedades tenga el nombre “enfermedads_controller”.

Disponemos de varios atributos que nos facilitan la estructuración de la aplicación.
El atributo $name se usa en PHP4, siendo opcional en PHP5, los atributos $components, $helpers y $uses permiten incluir funcionalidades extra en el controlador, siendo el más importante de los anteriores $uses, que permite incluir modelos, aparte del propio del controlador que se incluye por defecto.

En el ejemplo podemos ver que se incluye el atributo $paginate, el cual permite usar las funcionalidades de paginación.

Terminados los atributos podemos ver las funciones del controlador, como pueden ser: “index”, “view”, “add” o “modificar”. Podemos ver que cuando una función requiere mandar datos a una vista usa el método set y cuando necesita grabar datos utiliza el método save del modelo que estamos usando.

El método find que se puede observar en la función “view” ejecuta una consulta en la base de datos con los parámetros especificados, devuelve el primer valor de la tabla si no se le pasan parámetros. El método save por el contrario, efectúa una inserción en la tabla.

A continuación el código de enfermedads_controller:

```
class EnfermedadsController extends AppController {
	var $name = 'Enfermedads';
	var $uses = array('Enfermedad', 'Sintoma', 'Tratamiento');
	var $helpers = array('Form', 'Html', 'Javascript', 'Time');
	var $layout = 'ajax';

	var $components = array('RequestHandler');

	function beforeFilter(){
		if (!$this->RequestHandler->isAjax()){
			$this->layout = "default";
		}
	}

	function index() {
		$data = $this->paginate('Enfermedad');
		$this->set(compact('data'));
	}

	function view($id = null) {
```

```
		$parametros = array('conditions' => array('Enfermedad.id_e' => $id));
		$res = $this->Enfermedad->find('first', $parametros);
		$this->set('Sintomas',$this->Enfermedad->Sintoma->find('list',
array('fields'=>array('Sintoma.nombre','Sintoma.descripcion','Sintoma.id_s'))));
		$this->set('Tratamientos',$this->Enfermedad->Tratamiento->find('list',
array('fields'=>array('Tratamiento.id_t','Tratamiento.nombre','Tratamiento.descripcion')
)));
		$this->set('Enfermedad', $res);
	}

	function add() {
		$this->set('Sintomas',$this->Enfermedad->Sintoma->find('list',
array('fields'=>array('Sintoma.id_s','Sintoma.nombre'))));
		$this->set('Tratamientos',$this->Enfermedad->Tratamiento->find('list',
array('fields'=>array('Tratamiento.id_t','Tratamiento.nombre'))));
		if (!empty($this->data)) {
			if ($this->Enfermedad->save($this->data)) {
				$this->flash('Los datos de la Enfermedad se han guardado.',
'/enfermedads');
			}
		}
	}

	function modificar($id = null) {
		$this->Enfermedad->id = $id;
		$this->set('Sintomas',$this->Enfermedad->Sintoma->find('list',
array('fields'=>array('Sintoma.id_s','Sintoma.nombre'))));
		$this->set('Tratamientos',$this->Enfermedad->Tratamiento->find('list',
array('fields'=>array('Tratamiento.id_t','Tratamiento.nombre'))));
		if (empty($this->data)) {
			$this->data = $this->Enfermedad->read();
		} else {
			if ($this->Enfermedad->save($this->data)) {
				$this->layout = "default";
				$this->flash('Datosactualizados.',
'/enfermedads/view/'.$id);
			}
		}
	}

	function consultar() {
		if (!empty($this->data)) {
			$parametros=		array('fields'=>array('Enfermedad.id_e',
'Enfermedad.nombre'),'conditions'=>  array('Enfermedad.nombre_c  LIKE  '  =>  '%'.$this-
>data["Enfermedad"]["nombre_c"].'%','Enfermedad.nombre    LIKE    '    =>    '%'.$this-
>data["Enfermedad"]["nombre"].'%'));
			$res = $this->Enfermedad->find('list', $parametros);
			if (!empty($res)){
				$this->set("resultados",$res);
			}else{
```

```
                            $this->flash('Búsqueda           sin           resultados.',
'/enfermedads/consultar/');
                        }
                }
        }
}
```

En el código anterior podemos ver que se implementa una función llamada “BeforeFilter”, el objetivo de dicha función es identificar si la llamada se está efectuando mediante ajax, para, en caso contrario, hacer uso del layout por defecto.

4.5.3 HTML

El código HTML de la aplicación se encuentra en las plantillas correspondientes a cada función.

Se trata de los archivos con extensión “ctp”, en los cuales se encuentra el código HTML, con algo de código PHP insertado, para introducir los valores generados por la aplicación en la correspondiente plantilla.

Para generar el código de la forma más limpia posible, se ha hecho uso de algunas de las funcionalidades que incluye CakePHP, éstas permiten que mediante el uso de los objetos $ajax, $form o $html, y sus respectivas funciones; Se pueda generar el código HTML con el formato correcto.

Sirva como ejemplo el código correspondiente al formulario para la inserción de enfermedades.

```
echo $form->create('Enfermedad');
       echo $form->input('nombre_c',array('label' => 'Nombre Científico'));
       echo $form->input('nombre',array('label' => 'Nombre'));
       echo $form->input('aparicion',array('label' => 'Edad de Aparición'));
       echo $form->input('hereditaria',array('label' => 'Hereditaria','options'
=> array("SI" => "SI","NO" => "NO")));
       echo $form->end('Guardar');
```

De igual forma que el código anterior, se realizan todos los formularios HTML de la aplicación.

Capítulo 5: Pruebas y Usabilidad

5.1 Introducción

En este apartado se incluyen todas aquellas pruebas realizadas para verificar el correcto funcionamiento del sistema. En su mayoría se trata de inserciones de datos y su posterior visionado para verificar su correcto funcionamiento.

5.2 Información sobre el sistema

Como ya ha sido indicado en apartados anteriores, el sistema es una página Web desarrollada en php, que accede a una base de datos en MySQL para almacenar y recuperar la información.

Las pruebas a realizar se centran en su mayor parte en el correcto funcionamiento del almacenamiento y la recuperación de los datos. También se han realizado pruebas de usabilidad, cambiando el diseño de la Web para ofrecer una experiencia mejor al usuario.

5.2.1 Requisitos

Es necesario que el sistema almacene los datos correctamente, y que los devuelva de la misma forma y de tal manera que al usuario le resulte cómodo el visionado de dichos datos. Asimismo se debe evitar que los usuarios introduzcan datos erróneos en el sistema.

5.2.2 El sistema

El Framework CakePHP incluye en el modelo de datos la validación necesaria para cada uno. En caso de introducir datos erróneos se muestra en pantalla un mensaje de error indicando que se ha producido un fallo en la validación.

Si los datos introducidos son correctos, se procede a guardarlos y a enviar un mensaje de éxito.

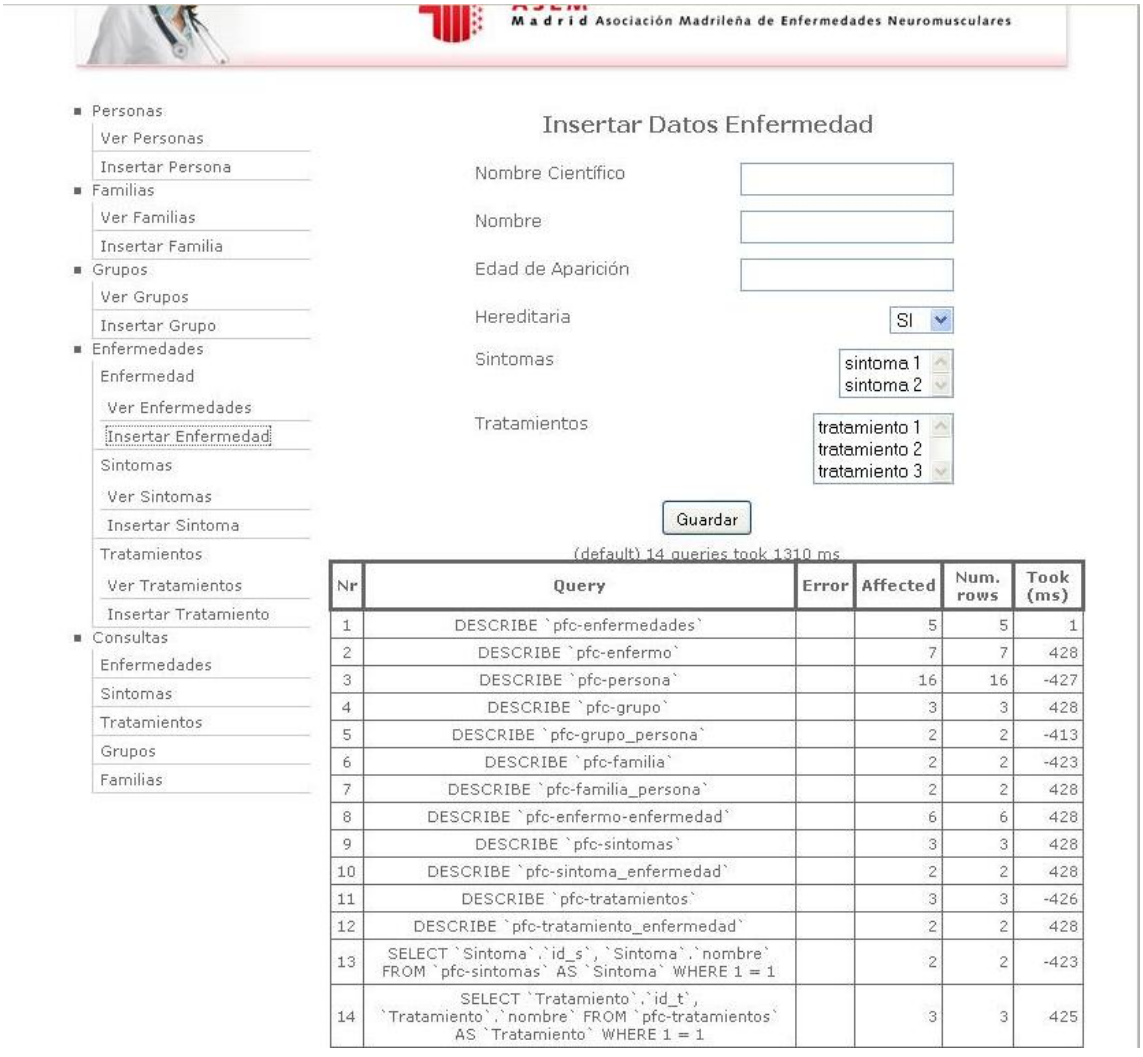
CakePHP incluye la posibilidad de visualizar las consultas efectuadas a la base de datos en cada función. De este modo podemos verificar que la aplicación hace lo que esperamos que haga.

5.2.3 Las pruebas

A continuación se muestran las pruebas realizadas para comprobar el funcionamiento de la aplicación. Debido a que hay una gran cantidad de formularios y páginas que requieren las pruebas, y puesto que de cara a esta memoria no aportan nada nuevo, solo se mostraran unos cuantos ejemplos de cada tipo.

Las pruebas realizadas consisten en la inserción, modificación y consulta de los datos, en cada uno de los formularios, así como la inserción de valores erróneos que permitan comprobar la validación de los datos.

5.2.3.1 Formulario inserción



Nr	Query	Error	Affected	Num. rows	Took (ms)
1	DESCRIBE `pfc-enfermedades`		5	5	1
2	DESCRIBE `pfc-enfermo`		7	7	428
3	DESCRIBE `pfc-persona`		16	16	-427
4	DESCRIBE `pfc-grupo`		3	3	428
5	DESCRIBE `pfc-grupo_persona`		2	2	-413
6	DESCRIBE `pfc-familia`		2	2	-423
7	DESCRIBE `pfc-familia_persona`		2	2	428
8	DESCRIBE `pfc-enfermo-enfermedad`		6	6	428
9	DESCRIBE `pfc-sintomas`		3	3	428
10	DESCRIBE `pfc-sintoma_enfermedad`		2	2	428
11	DESCRIBE `pfc-tratamientos`		3	3	-426
12	DESCRIBE `pfc-tratamiento_enfermedad`		2	2	428
13	SELECT `Sintoma`.`id_s`, `Sintoma`.`nombre` FROM `pfc-sintomas` AS `Sintoma` WHERE 1 = 1		2	2	-423
14	SELECT `Tratamiento`.`id_t`, `Tratamiento`.`nombre` FROM `pfc-tratamientos` AS `Tratamiento` WHERE 1 = 1		3	3	425

Podemos ver que al cargar el formulario de inserción, la aplicación efectúa una serie de consultas, concretamente a las tablas de tratamientos y síntomas para así obtener los listados necesarios.

5.2.3.2 Mensaje de éxito

Los datos de la Enfermedad se han guardado.

(default) 22 queries took 530 ms

Nr	Query	Error	Affected	Num. rows	Took (ms)
1	DESCRIBE `pfc-enfermedades`		5	5	5
2	DESCRIBE `pfc-enfermo`		7	7	5
3	DESCRIBE `pfc-persona`		16	16	5
4	DESCRIBE `pfc-grupo`		3	3	5
5	DESCRIBE `pfc-grupo_persona`		2	2	429
6	DESCRIBE `pfc-familia`		2	2	5
7	DESCRIBE `pfc-familia_persona`		2	2	5
8	DESCRIBE `pfc-enfermo-enfermedad`		6	6	5
9	DESCRIBE `pfc-sintomas`		3	3	5
10	DESCRIBE `pfc-sintoma_enfermedad`		2	2	4
11	DESCRIBE `pfc-tratamientos`		3	3	5
12	DESCRIBE `pfc-tratamiento_enfermedad`		2	2	16
13	SELECT `Sintoma`.`id_s`, `Sintoma`.`nombre` FROM `pfc-sintomas` AS `Sintoma` WHERE 1 = 1		2	2	1
14	SELECT `Tratamiento`.`id_t`, `Tratamiento`.`nombre` FROM `pfc-tratamientos` AS `Tratamiento` WHERE 1 = 1		3	3	1
15	SELECT COUNT(*) AS `count` FROM `pfc-enfermedades` AS `Enfermedad` WHERE `Enfermedad`.`nombre` = 'enfermedad3'		1	1	1
16	SELECT COUNT(*) AS `count` FROM `pfc-enfermedades` AS `Enfermedad` WHERE `Enfermedad`.`nombre_c` = 'Enfermedad numero 3'		1	1	1
17	INSERT INTO `pfc-enfermedades` (`nombre_c`,`nombre`,`aparicion`,`hereditaria`) VALUES ('Enfermedad numero 3','enfermedad3',23,'NO')		1		28
18	SELECT LAST_INSERT_ID() AS insertID		1	1	0
19	SELECT `Pfc-sintomaEnfermedad`.`id_s` FROM `pfc-sintoma_enfermedad` AS `Pfc-sintomaEnfermedad` WHERE `Pfc-sintomaEnfermedad`.`id_e` = 3		0	0	1
20	INSERT INTO `pfc-sintoma_enfermedad` (`id_e`,`id_s`) VALUES (3,'1')		1		1
21	SELECT `Pfc-tratamientoEnfermedad`.`id_t` FROM `pfc-tratamiento_enfermedad` AS `Pfc-tratamientoEnfermedad` WHERE `Pfc-tratamientoEnfermedad`.`id_e` = 3		0	0	1
22	INSERT INTO `pfc-tratamiento_enfermedad` (`id_e`,`id_t`) VALUES (3,'3')		1		1

Si la inserción se ha realizado de forma correcta, se devuelve un mensaje de éxito y posteriormente se redirige al usuario al índice de enfermedades.

Al encontrarnos en modo de pruebas, podemos observar las consultas realizadas por el sistema para efectuar la inserción, en este caso vemos que el sistema comprueba en los pasos 15 y 16 que no haya enfermedades con el nombre indicado, y en los pasos siguientes realiza la inserción, recupera el identificador de la enfermedad y lo inserta en las tablas de relación entre enfermedades, síntomas y tratamientos.

5.2.3.3 Mensaje de fracaso

Si el usuario intenta introducir un dato inválido, por dejarlo en blanco o por ser un dato que ya existe, al darle al botón Guardar, el sistema le mostrará un mensaje de error en aquellos campos en los que se hayan introducido datos no válidos.

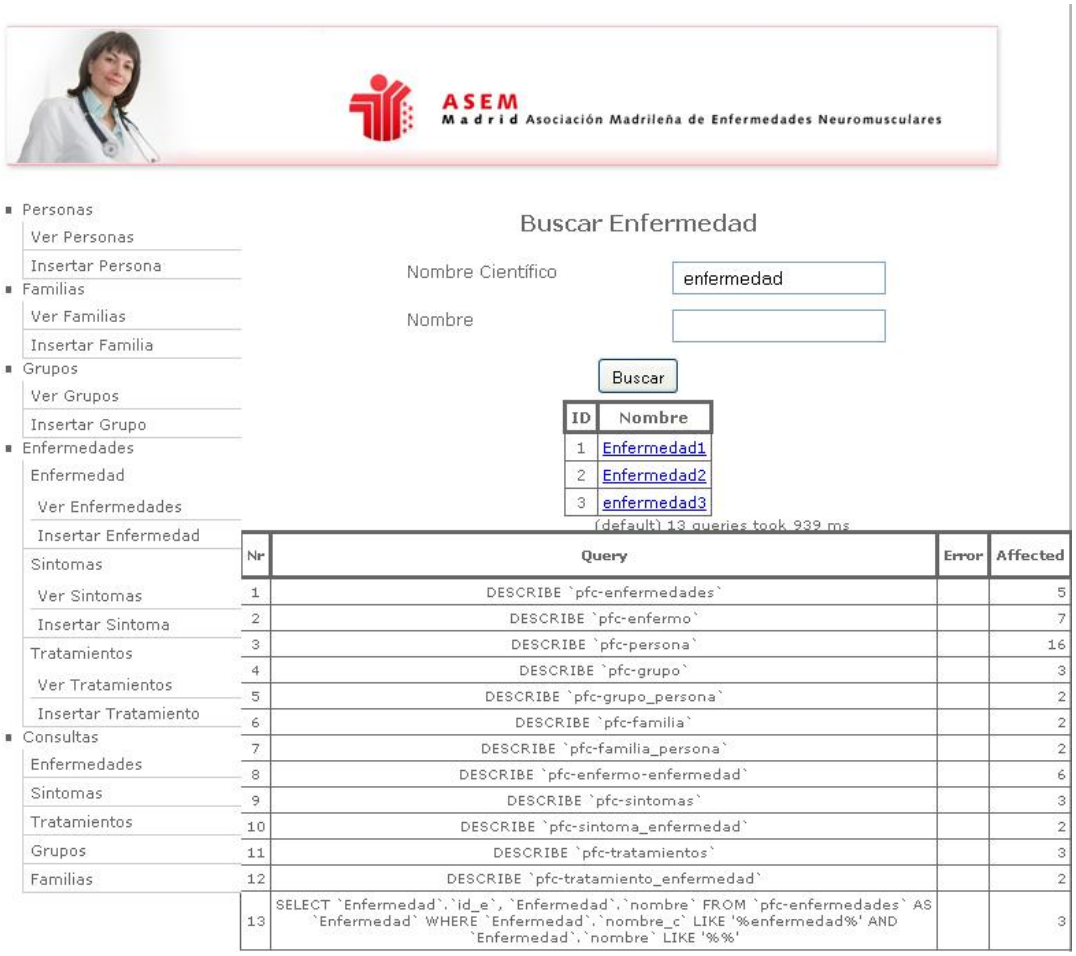
En el modo de pruebas, podemos ver como la aplicación ejecuta consultas en la tabla de enfermedades para comprobar si existe la enfermedad que se quiere introducir.

5.2.3.4 Formulario de consulta

Nr	Query	Error	Affected	Num. rows	Took (ms)
1	DESCRIBE `pfc-enfermedades`		5	5	4
2	DESCRIBE `pfc-enfermo`		7	7	4
3	DESCRIBE `pfc-persona`		16	16	4
4	DESCRIBE `pfc-grupo`		3	3	431
5	DESCRIBE `pfc-grupo_persona`		2	2	4
6	DESCRIBE `pfc-familia`		2	2	4
7	DESCRIBE `pfc-familia_persona`		2	2	4
8	DESCRIBE `pfc-enfermo-enfermedad`		6	6	4
9	DESCRIBE `pfc-sintomas`		3	3	4
10	DESCRIBE `pfc-sintoma_enfermedad`		2	2	4
11	DESCRIBE `pfc-tratamientos`		3	3	4
12	DESCRIBE `pfc-tratamiento_enfermedad`		2	2	4

Cada modelo del sistema tiene su formulario de consulta, cuyo aspecto es similar al que se puede ver arriba. Como se puede ver, la aplicación no efectúa consultas previas.

5.2.3.5 Muestreo de resultados



Los resultados de las búsquedas se muestran en la misma página que el formulario, de modo que el usuario pueda realizar tantas búsquedas como desee sin necesidad de cambiar de página.

En algunas de las consultas puede verse como hipervínculo el nombre que se ha buscado, en esos casos al pulsar sobre dicho nombre se verá la vista detallada del modelo.

En el modo de pruebas podemos ver la consulta SELECT que se ejecuta. La búsqueda devuelve un listado de resultados si los criterios de búsqueda lo permiten.

5.2.3.6 Fallo en la búsqueda

Búsqueda sin resultados.

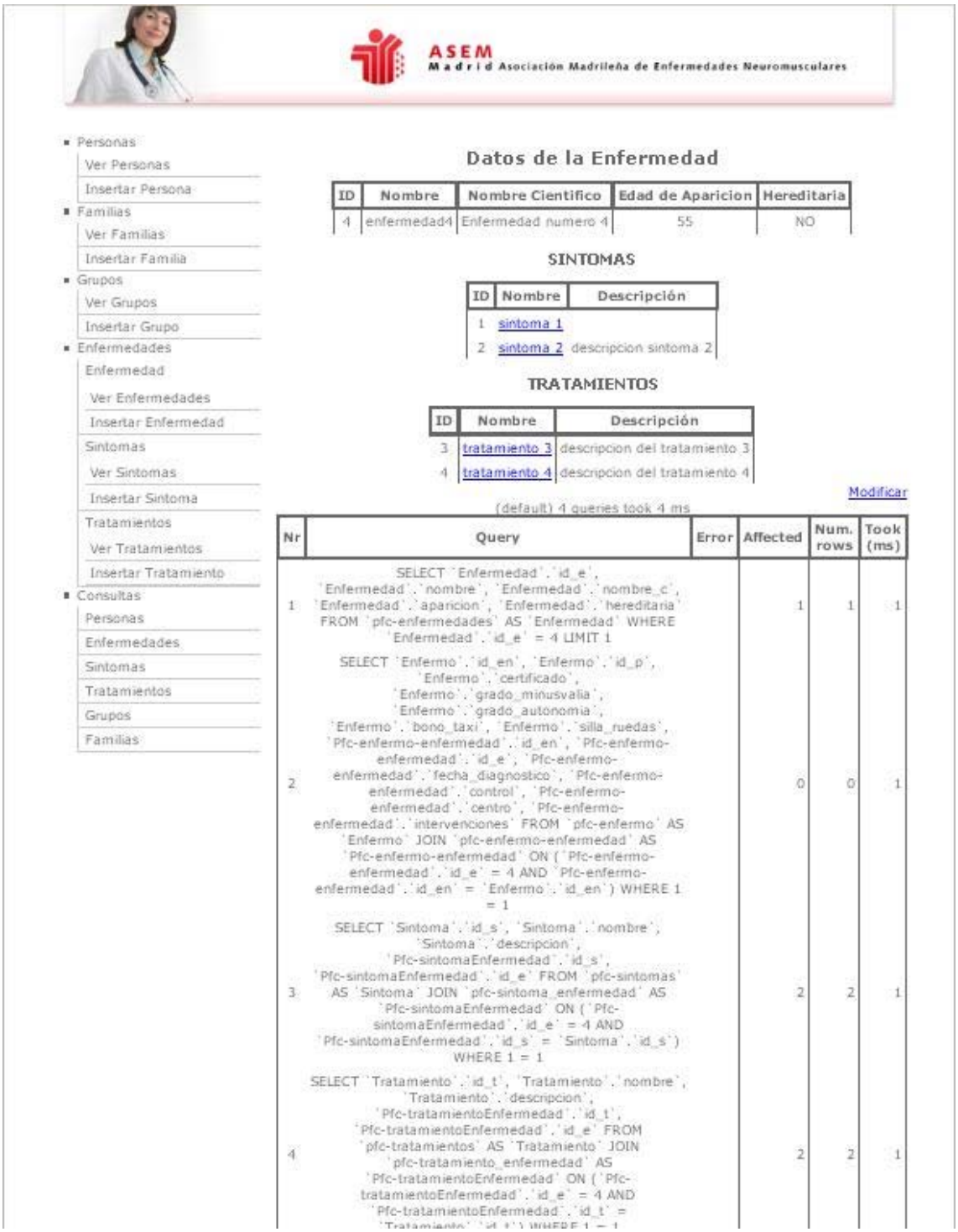
(default) 1 query took 1 ms

Nr	Query	Error	Affected	Num. rows	Took (ms)
1	SELECT `Enfermedad`.`id_e`, `Enfermedad`.`nombre` FROM `pfc-enfermedades` AS `Enfermedad` WHERE `Enfermedad`.`nombre_c` LIKE '%asd%' AND `Enfermedad`.`nombre` LIKE '%%'		0	0	1

En caso de que no exista ningún valor que cumpla con los criterios de búsqueda, se devuelve un mensaje de error.

En el modo de pruebas podemos ver la consulta que ha devuelto ese error.

5.2.3.7 Vista en detalle



En esta vista podemos observar, las características detalladas de una entidad concreta. En el caso de tratarse de enfermedades, veremos sus síntomas y tratamientos asociados.

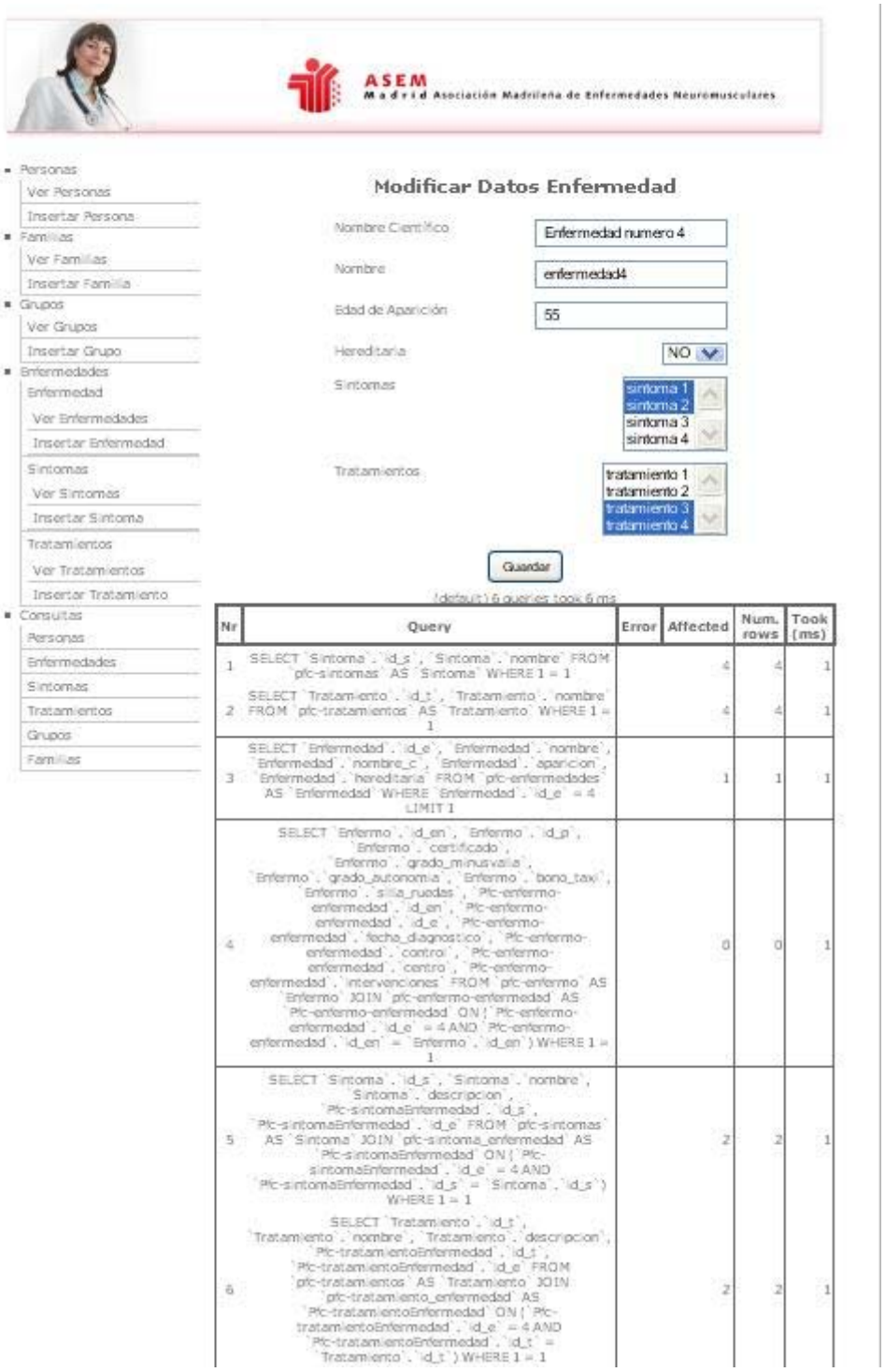
Si se trata de la vista detallada de un síntoma o un tratamiento, veremos las enfermedades a las que se encuentra asignado.

En caso de tratarse de grupos o familias, veremos las personas que pertenecen a los mismos, y en el detalle de personas, veremos a qué grupo pertenece.

En la esquina inferior derecha se puede ver el enlace “Modificar”, dicho enlace nos lleva al formulario de modificación correspondiente al modelo actual.

El modo de pruebas nos permite ver las consultas que nos han devuelto dichos datos.

5.2.3.8 Modificar Datos



En el formulario de modificación de datos, se cargarán los datos actuales pertenecientes al valor que deseamos modificar. Posteriormente, será posible realizar la modificación de los datos, verificándose los mismos, de igual forma que en la inserción.

En el modo de pruebas, podemos observar cómo se ejecutan las consultas que obtienen los valores actuales de los distintos parámetros.

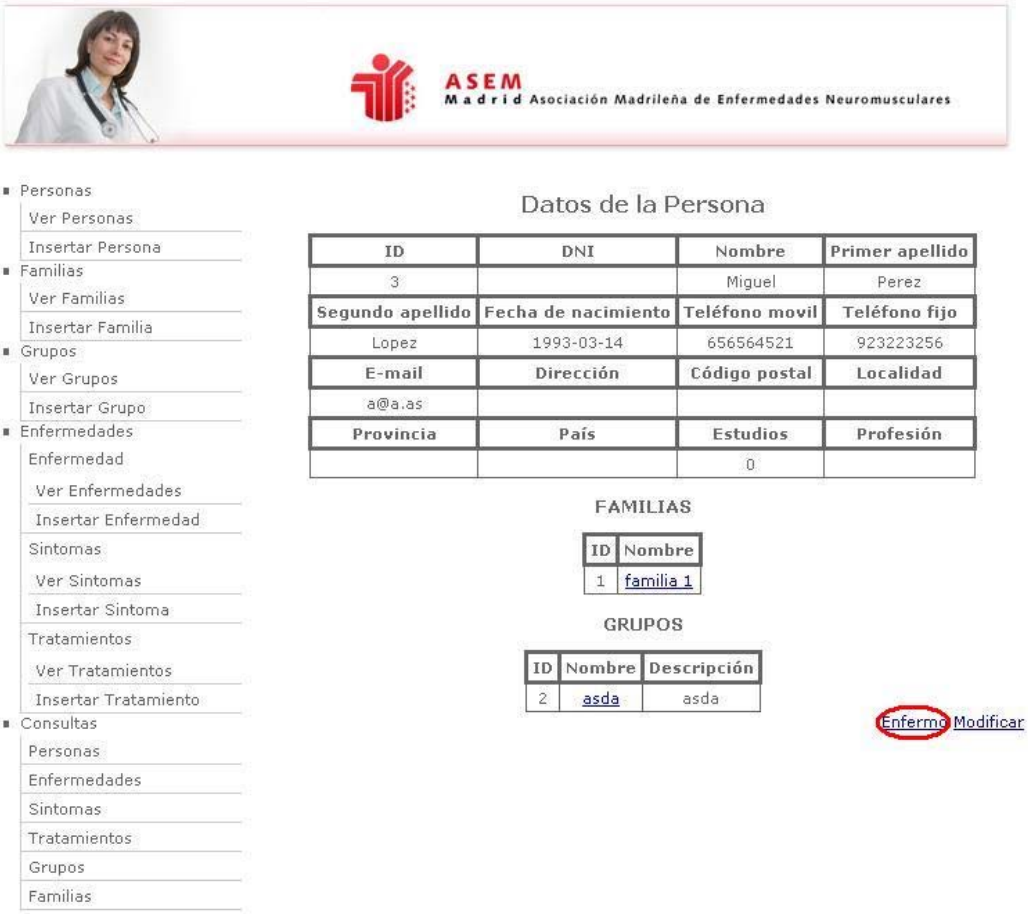
5.2.3.9 Borrar Datos

Para el borrado de datos solo tenemos que pulsar el botón correspondiente al lado del botón modificar. Tras el borrado aparecerá un mensaje de confirmación en la página.

Podemos observar, como la aplicación buscar todas las relaciones en las que el tratamiento se encuentra presente, y en este caso al no encontrar ninguna procede a borrar el tratamiento.

5.2.3.10 Caso especial: Enfermo

La inserción de los datos del enfermo, es un caso aparte en la aplicación, debido a que para poder insertar los datos de un enfermo, es necesario previamente que se hayan insertado los datos de una persona. En la vista de detalle de la persona podemos observar el enlace a Enfermo, como se muestra a continuación:



La aplicación comprueba la existencia de los datos del enfermo, en caso de no existir, muestra un mensaje de error, indicando que esa persona no está registrada como enfermo y mostrando posteriormente el formulario de inserción de datos del enfermo.

En caso de que el enfermo ya se encuentre registrado, la aplicación mostrará la vista en detalle de los datos del enfermo

Capítulo 6: Conclusiones y Futuros desarrollos

6.1 Objetivos

Los objetivos que se habían marcado en un principio eran los siguientes:

- El principal objetivo de este proyecto es crear la estructura de la base de datos, esta estructura debe ser adaptable a futuros cambios y ampliaciones en función de las necesidades de la organización, también es necesario que la aplicación sea flexible, no se dispondrá siempre de todos los datos.

Este objetivo ha sido cumplido, desarrollando una base de datos en MySQL, muy flexible y fácil de modificar en caso de cualquier necesidad.

- Elección de las tecnologías y herramientas a utilizar para el desarrollo. Las tecnologías finalmente empleadas en el proyecto se han indicado en el apartado de Estado de la cuestión. Sobre todo se ha procurado que fueran de libre distribución y de uso mayoritario, de forma que sea fácilmente modificable en un futuro.

Todas las tecnologías usadas en la realización del proyecto han sido de uso libre, evitando de ese modo un aumento de los costes.

- Implementar un interfaz de acceso que permita a los usuarios interactuar de forma sencilla con la base de datos, de este modo permitiremos que un usuario no cualificado pueda introducir información en el sistema, siempre manteniendo la estructura de nuestra base de datos.

Se ha diseñado e implementado un interfaz completo para la aplicación. Ha sido pensado de forma que sea lo más simple y cómodo posible para el usuario.

- Llevar a cabo las pruebas que permitan comprobar que la aplicación se encuentra en pleno funcionamiento.

Tras realizar el desarrollo se han llevado a cabo todas las pruebas necesarias para comprobar que el funcionamiento de la aplicación es el esperado.

- Aplicar los conocimientos adquiridos durante la carrera en el desarrollo de aplicaciones y bases de datos.

Gracias a la base de conocimientos obtenidos durante la carrera, el diseñado e implementación de la base de datos ha sido llevado a cabo de forma más óptima.

6.1.1 Conocimientos

En un principio el diseño de una nueva base de datos era imprescindible para que la información pudiese ser ordenada, durante el proceso de diseño de la misma, surgieron casos a tener en cuenta que complicaron un poco la tarea, pero que al mismo tiempo permitieron ampliar mis conocimientos sobre las bases de datos, al contemplar casos nuevos que no estaban previstos en el anterior diseño.

Posteriormente durante el desarrollo de la propia aplicación, me he encontrado con la oportunidad de hacer uso de características del lenguaje php que me eran desconocidas al momento de empezar con este proyecto. Una de las características que más me ha llamado la atención es la orientación a objetos, dado que mi experiencia previa con php se limitaba a aplicaciones estructuradas con código php embebido en el html.

Del mismo modo y gracias al framework CakePHP, he podido hacer uso del modelo vista-controlador (MVC), gracias a dicho modelo y al framework utilizado, el rendimiento durante la programación de la aplicación ha aumentado de forma sorprendente, ayudando a centrarte exclusivamente en los datos que deseas obtener, y despreocupandote de cómo los obtendrás.

Debido a la necesidad de hacer que la aplicación fuera amigable para el usuario, he necesitado hacer uso de las hojas de estilo en cascada (CSS), han sido una gran ayuda para cumplir de forma completa con el modelo vista-controlador, y al mismo tiempo han resultado un enorme quebradero de cabeza por la diferente interpretación que hacen los navegadores de las CSS.

6.2 Dificultades encontradas

En este apartado se indican las mayores dificultas que se han podido encontrar durante el desarrollo del proyecto.

6.2.1 Diseño de la BBDD

La mayor dificultad del proyecto se ha encontrado en el diseño de la base de datos del mismo. Debido a las especificaciones dadas por la ONG se han producido algunas complicaciones en

el diseño, uno de los mayores inconvenientes fue la necesidad de contemplar la posibilidad de personas con el mismo nombre y DNI. Dicha especificación no resultaba complicada de cara al diseño de la base de datos en sí, pero hacía complicado su uso al impedir identificar de forma unívoca a cada persona introducida en la base de datos mediante el DNI.

Al mismo tiempo también ha sido necesario tener en cuenta que un niño puede no tener DNI, de forma que si antes no podía ser identificador único, ahora es necesario que sea posible que tenga valor nulo.

Finalmente se ha optado por incluir como obligatorio el campo email, que serviría junto al nombre y el primer apellido para identificar a una persona de cara al usuario. De manera interna la base de datos simplemente identifica a la persona con un identificador único.

6.2.2 Implementación

Durante la fase de implementación me encontré con ciertas dificultades, debido al diseño inicial de la aplicación, en el que se mezclaban el código HTML con el código PHP, al estar este último incrustado en el primero.

Dicho inconveniente fue subsanado al hacer uso del Framework CakePHP, que permitió separar de forma clara el diseño de la lógica de la aplicación, pero ha ampliado los tiempos del proyecto ya que ha sido necesario aprender a hacer uso de las ventajas del framework.

6.3 Futuro del proyecto

El futuro del proyecto pasaría por ampliar las funcionalidades de la aplicación, permitiendo realizar consultas más complejas. También sería útil la inclusión de un control de usuarios que permitiera que la aplicación fuera accesible desde Internet de forma segura. Los grupos de usuarios ya creados podrían ser de gran utilidad en esa función.

Existen varias funcionalidades que debido a su mayor complejidad no han sido incluidas en la aplicación inicial, pero que si podrían ser añadidas en desarrollos futuros. Alguna de esas funcionalidades es el control de las cuotas de los miembros, que podría gestionarse desde la aplicación.

Anexo 1: Manual de Usuario

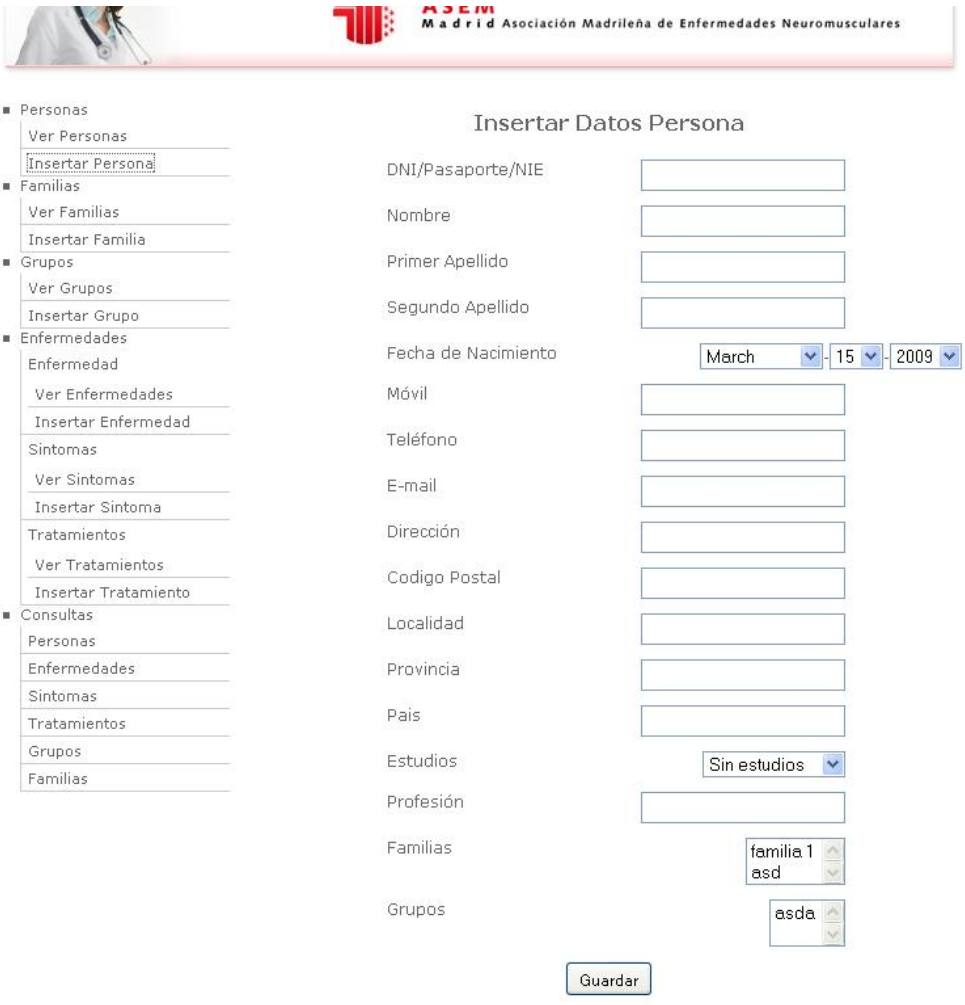
1.1 Introducción

El siguiente manual va dirigido al usuario de la aplicación web, se entiende que es una persona que no tiene conocimientos informáticos y por lo tanto se tratará de un manual lo más sencillo posible.

1.1.1 Inserción de datos

Existen varios formularios de inserción de datos, en dichos formularios aparecen indicados los campos que se necesitan completar. El programa indicará si falta algún dato que se considere necesario para el correcto funcionamiento de la aplicación.

Ejemplo de formulario de inserción:



1.1.2 Visualización de datos

Los datos se muestran a partir de un enlace que se puede encontrar en el índice de los datos o en el formulario de consulta.

El índice muestra todas las entradas de la base de datos para un modelo concreto.

Al pulsar en el nombre se accede a la vista de detalle, desde la cual es posible modificar los datos.

1.1.3 Modificación de datos

La modificación de los datos se realiza de forma muy similar a la inserción. Los formularios de modificación tienen un aspecto similar a los anteriores.

Ejemplo de formulario de modificación:

1.1.4 Consulta de datos

Todas las consultas se encuentran agrupadas en el apartado “Consultas” en el menú. No es necesario saber un dato completo para que se muestre el resultado, basta con saber una parte. El resultado se muestra en una tabla desde la cual se puede ir a la vista en detalle de la información buscada.

Ejemplo de formulario de búsqueda:

1.2 Instalación

La instalación de esta aplicación se realiza de forma muy simple. Al tratarse de una página web, basta con disponer de un servidor web que tenga soporte para PHP y MySQL.

Una vez creadas las tablas en la base de datos solo es necesario copiar los archivos de la web a la carpeta local. Con eso la aplicación se encontrará operativa.

Anexo 2: Manual del Programador

2.1 Introducción

El siguiente manual va dirigido a futuros programadores de la aplicación web, se entiende que es una persona que tiene conocimientos de PHP, HTML, CSS y MySQL.

2.1.1 Estructura de la aplicación

La aplicación se encuentra ordenada en la configuración estándar para CakePHP, de modo que todos los datos que pueden necesitar modificación, se encuentran en la carpeta app, de la raíz del proyecto.

Una vez dentro, los archivos referidos a los distintos modelos de datos se encuentran en la carpeta models, estos archivos contienen la validación de datos y las relaciones entre modelos.

También es relevante la carpeta controllers, que como su nombre indica, contiene los controladores de la aplicación, o lo que es lo mismo, la lógica con que se tratan los datos.

La carpeta views contiene las plantillas que corresponden a cada función de un controlador, una función puede tener varias vistas, depende del controlador cual se muestra en cada momento.

Por otra parte la carpeta webroot, incluye las hojas de estilo, javascript, imágenes, etc. Todos aquellos ficheros externos a la aplicación y que puedan ser requeridos por la misma.

Finalmente la carpeta config, contiene los ficheros de configuración de la aplicación. Los ficheros presentes en esta carpeta son de vital importancia, ya que controlan gran variedad de aspectos.

El fichero “database.php” define las bases de datos que se van a utilizar en el proyecto, CakePHP permite hacer uso de varias bases de datos distintas de forma sencilla.

El fichero “core.php” configura el modo en que se encuentra la aplicación, con un modo de producción y 3 distintos para desarrollo, también establece los tiempos de cache, y la duración de las cookies en caso de utilizarse.

Para terminar, el fichero “routes.php” se encarga de controlar la url desde la que se accede a la aplicación, y de redirigir cada url a la función deseada, en nuestro caso, la raíz del proyecto está dirigida a una clase llamada inicio, cuya única función consiste en mostrar el mensaje de bienvenida a la aplicación.

2.1.2 Modificar aplicación

Siguiendo la estructura propuesta anteriormente, la modificación de la aplicación es simple. Si se pretende añadir nueva funcionalidad a los modelos existentes, solo hay que añadir una función nueva al controlador apropiado y crear la template asociada a dicha función.

En caso de que se desee modificar la base de datos, solo sería necesario modificar el modelo de la aplicación si se cambiara el nombre de algún campo con control de validación, o si se añadiera algo que requiriera validación.

Si el cambio que se pretende realizar, afecta únicamente la estructura en la que se muestran los datos, bastaría con modificar el template correspondiente a la salida que se desea alterar.

2.1.3 Modificación del interfaz de la aplicación

El aspecto de la aplicación se encuentra definido mediante hojas de estilo en cascada (CSS). Si se quisiera realizar algún cambio en el aspecto de la aplicación bastaría con realizarlo en el fichero “estilo.css”.

Sin embargo aquellos cambios que requieran modificar el código HTML de la aplicación tendrán que acceder al fichero php correspondiente a dicho módulo.

2.1.4 Modificación de la estructura de la base de datos

La base de datos está realizada en MySQL, la modificación de la misma se puede realizar fácilmente mediante el uso de aplicaciones como phpmyadmin. Es recomendable tener en cuenta, que los cambios pueden suponer una alteración del modelo de datos, y en ese caso podría hacer necesarios varios cambios en la aplicación.

Anexo 3: Planificación

En este apartado se muestra la información referida a la planificación, asignación de recursos, costes, etc. De las tareas del proyecto. Mediante esta información, será posible conocer una estimación del tiempo total de ejecución del proyecto y la rentabilidad económica del mismo.

Para dicha planificación se ha hecho uso de la aplicación GanttProject (http://ganttproject.biz/).

3.1 Tareas del diagrama de Gantt

En este apartado se van a mostrar las tareas definidas en el diagrama Gantt durante la planificación del proyecto. Se han agrupado las tareas para que sea posible verlas de una forma más clara.

Las tareas marcadas en negrita son aquellas que engloban varias tareas.

1 PFC
- **1.1 ANTEPROYECTO**
  - 1.1.1 Documentación
  - 1.1.2 Análisis de herramientas
  - 1.1.3 Estudio del arte
  - 1.1.4 Estudio de viabilidad
  - 1.1.5 Recogida de requisitos
- **1.2 DISEÑO DEL SISTEMA**
  - 1.2.1 Diseño de la base de datos
  - 1.2.2 Diseño de la estructura de la aplicación
  - 1.2.3 Diseño de las páginas
- **1.3 IMPLEMENTACIÓN**
  - **1.3.1 Entorno de desarrollo**
    - 1.3.1.1 Plataforma servicios web
    - 1.3.1.2 Herramientas de desarrollo
  - **1.3.2 Base de datos**
    - 1.3.2.1 Creación de la base de datos
  - **1.3.3 Aspecto gráfico**
    - 1.3.3.1 Maquetación de las páginas
  - **1.3.4 Funcionalidad**
    - 1.3.4.1 Programación de las páginas
- **1.4 PRUEBAS**
  - 1.4.1 Realización de las pruebas
  - 1.4.2 Corrección de fallos inicial
  - 1.4.3 Pruebas de la ONG
  - 1.4.4 Corrección de fallos final
- **1.5 MEMORIA**
  - 1.5.1 Recopilación de la información producida
  - 1.5.2 Redacción

3.2 Recursos

Los recursos son aquellos “elementos” necesarios para la realización del proyecto. En el caso de un proyecto informático como este, dichos recursos representan al personal encargado de llevar a cabo la aplicación.

Los recursos del proyecto son los siguientes:

- **Analista**: Persona encargada de llevar a cabo el análisis y diseño del proyecto, así como de llevar a cabo la documentación del mismo y las pruebas de funcionamiento.
- **Diseñador**: Persona encargada de realizar el diseño de las páginas para hacer que estas sean más amigables de cara al usuario.
- **Desarrollador**: Persona que lleva a cabo la implementación de la base de datos y de la programación de la aplicación.

3.3 Calendario

Los recursos tienen asignado un calendario laboral con los días de trabajo y su horario. En este proyecto se ha utilizado un calendario de trabajo de 5 días a la semana (lunes a viernes) y 8 horas (8:00 – 12:00 y 13:00 – 17:00), incluyendo los días festivos y teniendo dos semanas de vacaciones en Navidad y en Agosto.

3.4 Costes de los recursos

Cada uno de los recursos tiene especificado un coste que permite hacer una aproximación del coste total del proyecto. En el caso de este proyecto, se trata de los costes por hora de trabajo.

El coste de los recursos del proyecto es el siguiente:

Recurso	Coste
Analista	12 €
Diseñador	10 €
Desarrollador	8 €

3.5 Duración de las tareas

Las tareas reciben una asignación de tiempo, en el cual se trata de aproximar su duración real. La medida son días laborables.

Tareas	Duración
1.1 ANTEPROYECTO	83
1.1.1 Documentación	20
1.1.2 Análisis de herramientas	13
1.1.3 Estudio del arte	25
1.1.4 Estudio de viabilidad	10
1.1.5 Recogida de requisitos	15
1.2 DISEÑO DEL SISTEMA	55
1.2.1 Diseño de la base de datos	30
1.2.2 Diseño de la estructura de la aplicación	20
1.2.3 Diseño de las páginas	5
1.3 IMPLEMENTACIÓN	82
1.3.1 Entorno de desarrollo	20
1.3.1.1 Plataforma servicios web	10
1.3.1.2 Herramientas de desarrollo	10
1.3.2 Base de datos	20
1.3.2.1 Creación de la base de datos	20
1.3.3 Aspecto gráfico	20
1.3.3.1 Maquetación de las páginas	20
1.3.4 Funcionalidad	40
1.3.4.1 Programación de las páginas	40
1.4 PRUEBAS	60
1.4.1 Realización de las pruebas	20
1.4.2 Corrección de fallos inicial	5
1.4.3 Pruebas de la ONG	30
1.4.4 Corrección de fallos final	5
1.5 MEMORIA	60
1.5.1 Recopilación de la información producida	30
1.5.2 Redacción	30

3.6 Costes de las tareas

Para calcular los costes de las tareas, primero se unen los recursos con cada tarea del proyecto y asignar a cada recurso un porcentaje de esfuerzo que indica el tiempo que dedican a dicha tarea, para después poder calcular el coste de la tarea en base al tiempo necesario para completar la tarea por el/los recursos relacionado/s con ésta.

En este proyecto, los recursos se han relacionado con las tareas de la siguiente forma:

Tareas	Recursos [porcentaje esfuerzo]
1.1.1 Documentación	Analista [100%]
1.1.2 Análisis de herramientas	Analista [100%]
1.1.3 Estudio del arte	Analista [100%]
1.1.4 Estudio de viabilidad	Analista [100%]
1.1.5 Recogida de requisitos	Analista [100%]
1.2.1 Diseño de la base de datos	Analista [50%], Programador [50%]
1.2.2 Diseño de la estructura de la aplicación	Analista [50%], Programador [50%]
1.2.3 Diseño de las páginas	Diseñador [100%]
1.3.1.1 Plataforma servicios web	Programador [100%]
1.3.1.2 Herramientas de desarrollo	Programador [100%]
1.3.2.1 Creación de la base de datos	Programador [100%]
1.3.3.1 Maquetación de las páginas	Programador [100%]
1.3.4.1 Programación de las páginas	Programador [100%]
1.4.1 Realización de las pruebas	Analista [100%]
1.4.2 Corrección de fallos inicial	Analista [50%], Programador [50%]
1.4.3 Pruebas de la ONG	
1.4.4 Corrección de fallos final	Programador [100%]
1.5.1 Recopilación de la información producida	Analista [100%]
1.5.2 Redacción	Analista [100%]

A partir de los datos anteriores podemos concluir el coste total de cada tarea.

Tareas	Coste (en euros)
1.1.1 Documentación	1920
1.1.2 Análisis de herramientas	1248
1.1.3 Estudio del arte	2400
1.1.4 Estudio de viabilidad	960
1.1.5 Recogida de requisitos	1440
1.2.1 Diseño de la base de datos	2400
1.2.2 Diseño de la estructura de la aplicación	1600
1.2.3 Diseño de las páginas	400
1.3.1.1 Plataforma servicios web	640
1.3.1.2 Herramientas de desarrollo	640

1.3.2.1 Creación de la base de datos	1280
1.3.3.1 Maquetación de las páginas	1280
1.3.4.1 Programación de las páginas	2560
1.4.1 Realización de las pruebas	1920
1.4.2 Corrección de fallos inicial	800
1.4.3 Pruebas de la ONG	0
1.4.4 Corrección de fallos final	320
1.5.1 Recopilación de la información producida	2880
1.5.2 Redacción	2880

3.7 Duración y presupuesto del proyecto

En este apartado se especifican los detalles de la planificación, la duración de las tareas y los costes finales del proyecto.

3.7.1 Duración del proyecto

Para calcular la duración solo es necesario realizar la suma de los días laborables. Tras realizar la planificación en el programa el resultado es que la duración del proyecto es de 340 días laborables. Desde el 09/03/2007 al 09/07/2008.

3.7.2 Presupuesto del proyecto

Para calcular el presupuesto del proyecto, es necesario sumar al coste de las tareas aquellos recursos adicionales que puedan haber sido necesitados para las mismas, como puede ser un servidor.

Tareas	Coste (En Euros)
1.1 ANTEPROYECTO	7968
1.2 DISEÑO DEL SISTEMA	4400
1.3 IMPLEMENTACIÓN	6400
1.4 PRUEBAS	3040
1.5 MEMORIA	5760
Total:	27568

Al resultado anterior debemos sumarle el coste de los recursos que no se han contabilizado, en nuestro caso se trata de un único ordenador que hace las veces de servidor y de cliente

durante las pruebas. El valor aproximado de ese recurso son 1000€ más que se sumarán al presupuesto.

Dado que todo el software utilizado ha sido de libre distribución, no es necesario sumar ningún otro coste al total. De manera que:

El coste total del proyecto asciende a 28568€

BLIBIOGRAFIA

(1) Andrés, M. M. *Historia de los sistemas de bases de datos.* 12 de 02 de 2001. http://www3.uji.es/~mmarques/f47/apun/node6.html (último acceso: 2008).

(2) Angel Cobo, Patricia Gómez, Daniel Pérez, Rocío Rocha. *PHP y MySQL-tecnologias para el desarrollo de aplicaciones web.* Diaz de Santos.

(3) Cake Development Corporation. *The CakePHP Framework Experts.* http://cakedc.com/ (último acceso: 2009).

(4) CakePHP ORG. *CakePHP: the rapid development php framework.* http://cakephp.org/ (último acceso: 2009).

(5) Date, C. J. *Introducción a los Sistemas de Bases de Datos.* Pearson Prentice Hall.

(6) Desarrollo Web. *Desarrollo Web.* http://www.desarrolloweb.com/articulos/estructura-modelo-relacional.html (último acceso: 2008).

(7) La Web del Programador. *La Web del Programador.* http://www.lawebdelprogramador.com (último acceso: 2008).

(8) Michael Kruckenberg, Jay Pipes. *Pro MySQL.* Apress.

(9) Minera, Francisco. *AJAX.* Gradi.

(10) Minera, Francisco. *PHP Master.* Gradi.

(11) MySQL AB. *MySQL 5.0 Reference Manual.* http://dev.mysql.com/doc/refman/5.0/es/ (último acceso: 2008).

(12) Raghu Ramakrishnan, Johannes Gehrke. *Database management systems.* McGraw-Hill Professional, 2003.

(13) S. Esakkirajan, S. Sumathi. *Fundamentals of Relational Database Management Systems.* Springer.

(14) Symfony Team. *Symfony | Web PHP Framework.* http://www.symfony-project.org/ (último acceso: 2009).

(15) The PHP Group. *PHP: Hypertext Preprocessor*. http://es.php.net/ (último acceso: 2008,2009).

(16) Universidad Pontificia Javeriana. *Historia de las bases de datos en Ciencia de la Información.* http://recursostic.javeriana.edu.co/wiki/index.php/Historia_de_las_bases_de_datos_en_Ciencia_de_la_Información (último acceso: 2008).

(17) Wikimedia Foundation, Inc. *Wikipedia,la enciclopedia libre.* http://es.wikipedia.org (último acceso: 2008).

(18) Zend. *Zend.* http://www.zend.com/en/ (último acceso: 2009).

(19) World Wide Web Consortium. *World Wide Web Consortium.* http://www.w3.org/ (último acceso: 2008).